Volume 4, Issue 9, September 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Windows Azure: A Highly Available Storage of Cloud Service through Secured Channels

Ravi Sankar G*
Dept. of IT
[a] RGM College of Engineering and Technology
India

Rajasekhar Reddy V
Dept. of IT
RGM College of Engineering and Technology
India

Arun Babu P
Dept. of IT
RGM College of Engineering and Technology
India

Hara Gopal V P
Dept. of IT
RGM College of Engineering and Technology
India

Abstract— The main scope of this project is to understand how to best utilize different cloud computing services to get good performance for less cost. Microsoft Windows azure gained popularity in providing good cloud computing services hence is used here. This paper also deals with performance analysis of azure storage from different regions and performance between the regions. Cloud computing is best suited for small organizations which are worried about cost incurred in maintaining the infrastructure. Security is main concern of the cloud based computing and can be the future work of this project.

Keywords— Cloud, SaaS, PaaS, IaaS, Windows Azure, Computing, Google, Microsoft, Rachspace, Amazon

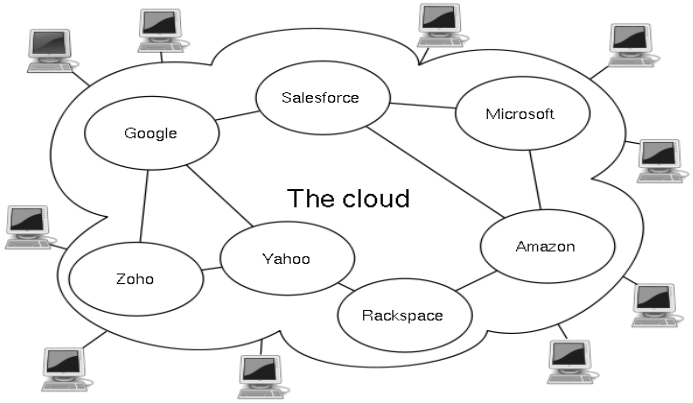
I. INTRODUCTION

Cloud computing is Internet based development and use of computer technology. In concept, it is a paradigm shift whereby details are abstracted from the users who no longer need knowledge of, expertise in, or control over the technology infrastructure "in the cloud" that supports them. It typically involves the provision of dynamically scalable and often virtualized resources as a service over the Internet.

The term cloud is used as a metaphor for the Internet, based on how the Internet is depicted in computer network diagrams and is an abstraction of the underlying infrastructure it conceals. Typical cloud computing services provide common business applications online that are accessed from a web browser, while the software and data are stored on the servers.

These services are broadly divided into three categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The name cloud computing was inspired by the cloud symbol that is often used to represent the Internet in flow charts and diagrams.

"Cloud Computing" refers to the use of Internet based computer technology for a variety of services. It is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet on a pay-for-use basis, at a fraction of the cost of provisioning a traditional Data Center based solution. All the costs associated with setting up a data center such as procuring a building, hardware, redundant power supply, cooling systems, upgrading electrical supply, and maintaining a separate Disaster Recovery site can be passed on to a third party vendor. Since the customer is charged only for computer services used, cloud computing costs are a fraction of traditional technology expenditures.



As a metaphor for the Internet, "the cloud" is a familiar cliché, but when combined with "computing," the meaning gets bigger and fuzzier. Some analysts and vendors define cloud computing narrowly as an updated version of utility computing: basically virtual servers available over the Internet. Others go very broad, arguing anything we consume outside the firewall is "in the cloud," including conventional outsourcing.

Cloud computing comes into focus only when we think about what IT always needs: a way to increase capacity or add capabilities on the fly without investing in new infrastructure, training new personnel, or licensing new software. Cloud computing encompasses any subscription-based or pay-per-use service that, in real time over the Internet, extends IT's existing capabilities.

II. CLOUD PROVIDERS

A service provider that offers customers storage or software services available via a private or public network. Usually, it means the storage and software is available for access via the Internet.

Infrastructure as a Service (IaaS) is a cloud model which allows organizations to outsource computing equipment and resources such as servers, storage, networking as well as services, such as load balancing and content delivery networks. The IaaS provider owns and maintains the equipment while the organization rents out the specific services it needs, usually on a "pay as you go" basis. Today, the question is less about whether or not to use IaaS services, but rather which providers to use. It's easy to think of IaaS as a commodity product, that all virtual machines are essentially the same regardless of where they reside. There is some truth in that, however the distinguishing features of each cloud provider, especially around management and specialized services, are what's important to administrators. When evaluating IaaS providers, it's important to look beyond the range of cloud services that are offered and consider management functions, monitoring tools, identity management, service level agreements and customer support.

IaaS Providers:

1 Amazon AWS

Amazon Web Services offers a full range of compute and storage offerings, including on-demand instances and specialized services such as Amazon Elastic Map Reduce (EMR) and Cluster GPU instances, as well as Elastic Block Storage (EBS) and high performance SSDs on the storage side. Additionally, the IaaS offers infrastructure services such as workflows, message passing, archival storage, in-memory caching services, search services, both relational and NoSQL databases and more.

2.Windows Azure

Despite the name, Windows Azure is not a Windows-only IaaS. The compute and storage services offered are typical of what you'll find in other IaaS providers, and administrators used to Microsoft platforms will find working with Windows Azure much easier. The IaaS offers ready access to virtual networks, service buses, message queues, and non-relational storage platforms as well.

3.Google Compute Engine

Google Compute Engine is well suited for big data, data warehousing, high performance computing and other analytics-focused applications. It is well integrated with other Google services, such as Google Cloud Storage, Google BigQeury and Google Cloud SQL. Although Google Compute Engine is still relatively new in the IaaS market, the fact that it runs on Google's global infrastructure, including the company's private global fiber network and high efficiency data centers sets it apart.

4.Rackspace Open Cloud

Rackspace offers core cloud computing services with a strong focus on customer service. Rackspace is one of the co-founders of OpenStack, which it uses for its cloud infrastructure, so you can run the same platform in-house if you decide to move to a private or hybrid cloud down the road. You can choose from a variety of operating systems, including Linux and Windows Server. And for an extra fee, you can easily create basic monitoring checks, like ping or HTTP checks.

5. IBM SmartCloud Enterprise

IBM SmartCloud Enterprise offers core compute and storage services in a 5-tier model along with an asset catalog of IBM and non-IBM software. The IaaS is ideal for enterprises managing a large number of developers and testers who need to deploy virtual machines and allocate storage as efficiently as possible. You can manage administrator and user roles, set limits on resources users can deploy and readily report on user activity.

Platform as a service (PaaS)

Platform as a service encapsulates a layer of software and provides it as a service that can be used to build higher-level services. There are at least two perspectives on PaaS depending on the perspective of the producer or consumer of the services:

- Someone *producing* PaaS might produce a platform by integrating an OS, middleware, application software, and even a development environment that is then provided to a customer as a service.
- Someone *using* PaaS would see an encapsulated service that is presented to them through an API. The customer interacts with the platform through the API, and the platform does what is necessary to manage and scale itself to provide a given level of service. Virtual appliances can be classified as instances of PaaS. A content switch appliance, for example, would have all of its component software hidden from the customer, and only an API or GUI for configuring and deploying the service provided to them.

PaaS offerings can provide for every phase of software development and testing, or they can be specialized around a particular area such as content management. Commercial examples of PaaS include the Google Apps Engine, which serves applications on Google's infrastructure. PaaS services such as these can provide a powerful basis on which to deploy applications, however they may be constrained by the capabilities that the cloud provider chooses to deliver.

PaaS Providers :

PaaS Provider: Engine Yard

Engine Yard is designed for web application developers using Ruby on Rails, PHP and Node.js who want to take advantage of cloud computing without the operations management responsibility. Engine Yard provides a set of services on top of Amazon AWS. In fact, Engine Yard runs its platform in the Amazon cloud, so the value of the PaaS rests more with orchestration and management than with providing software components. Engine Yard takes care of key operations tasks such as performing backups, managing snapshots, managing clusters, administering databases and load balancing.

Red Hat OpenShift

Red Hat OpenShift is based on open source applications and offers a wide variety of languages, databases and components. The PaaS is highly customizable and offered in three forms: OpenShift Online (a cloud-based hosting service), OpenShift Enterprise (a private PaaS that runs in your data center) and OpenShift Origin (the open source application hosting platform). OpenrShift automates system administration tasks such as virtual server provisioning, configuration and scaling and supports git repositories for code management.

Google App Engine

Google App Engine is designed for distributed web applications and developers using Java, Python, PHP and Go. The Java environment supports other languages that make use of the JRE and there is a SDK for each of the four main supported languages as well as a plugin for Eclipse. The PaaS offers managed infrastructure and runtime environments that are guaranteed to scale, but only if the applications fit the restrictions of Google App Engine. The Datastore, a transactional, schema-less data store

Heroku

Heroku supports Ruby, Python, Java, Scala, Cloture and Node.js. The platform provides abstract computing environments called dynos, which are virtualized Unix-style containers that run processes in isolated environments. Dynos are broadly typed as either web dynos (respond to HTTP requests) or worker dynos (respond to task requests in a queue). Heroku works best with applications that fit well into the Twelve Factor App methodology. Third party applications, called addons, are also available as services within the Heroku platform.

AppFog

AppFog is a multi-language, multi-framework PaaS that's a good option for multi-cloud deployments, including private clouds. AppFog supports Java, Ruby, PHP, Python, Node, Scala and Erlang and offers MySQL, PostgreSQL, Redis and RabbitMQ along with third party add-on services. It can be used as a DaaS with ClearDB, MongoHQ, MongoLab, Redis Cloud and Xeround. AppFog is based on the open source Cloud Foundry platform and supports Git, SVN and Mercurial for code management.

Caspio

Caspio is fundamentally different from the other PaaS providers on this list. It does not offer a fully functional software development environment; instead it focuses on bringing desktop database-like functionality to the cloud. Caspio is designed for creating basic databases, providing data entry forms and generating reports. It includes a point-and-click visual development tool with little, if any, coding required. Caspio applications are deployed by copying snippets of code to a web page or by accessing a URL to an application page. Caspio also comes with a set of preconfigured applications.

Software as a Service (SaaS)

SaaS is a new model of how software is delivered. SaaS refers to software that is accessed via a web browser and is paid on a subscription basis (monthly or yearly). Different from the traditional model where a customer buys a license to software and assumes ownership for its maintenance and installation, SaaS presents significant advantages to the customer.

SaaS is faster and a cost effective way to getting implemented. There are no hardware, implementation or acquisition costs involved to run the application from the customer's side. It's the responsibility of the SaaS vendor (us) to manage and run the application with utmost security, performance and reliability.

Since customers pay a subscription, they have immediate access to the new features and functionality. Unlike traditional softwares where upgrades would happen once a year or once in 6 months (with the vendor coming to your office with a CD), the SaaS vendor continuously pushes new updates, fixes to the application, which is immediately accessible by the customer. This reduces the length of time it takes a customer to recognize value from the software.

Since the software application is delivered as a service, its important for the vendor to focus on customer service and experience. Since this is on a subscription model, the vendor is judged on a month-month basis and the pressure to innovate or risk losing business is greater.

SaaS can be used by Windows, Linux, or Mac users, providing true platform independence over the Internet.

SaaS Providers

SaaS Customer Service Providers

Mobile devices, social media and the growing expectations of customers are shifting the way support is handled. Cloud-based solutions promise to transform customer service from a cost center into a profit center, by enabling self-service and brand championship. Today there are dozens of SaaS customer support platforms, some of which extend into marketing tools. We review and compare three popular solutions: ZenDesk Plus, Get Satisfaction and Salesforce's Desk.com.

SaaS Office Suite Providers

Today's cloud office suites offer anywhere access and cost savings over traditional, desktop software. The solutions offer enterprises standardization, resilience, ease of deployment and more. We examine the key elements, differences, as well

as pros and cons of cloud-based office suites. We then provide you with a helpful comparison guide to Google Apps for Business, Office 365 and Zoho Office.

SaaS Project Management Providers

Project managers have a number of cloud-based options to manage projects, which can help streamline their tasks and improve collaboration. We evaluate the key features and buying considerations, like portfolio management and application integration. We also offer a side-by-side comparison of four enterprise project management tools in the cloud: Clarizen, GroupCamp, Planzone and Zoho Projects.

SaaS Help Desk Providers

SaaS help desk solutions can streamline support administration, reduce infrastructure costs, and improve the way you communicate with your internal and external customers by extending support to different types of users. When evaluating cloud-based help desk providers, like JitBit, FreshDesk, ZenDesk and Desk.com consider the ticket management, knowledgebase support, reporting, customization and integration.

SaaS Network Monitoring Buyer's Guide

Cloud-based IT solutions are growing in popularity and the networking space is no short of them. Today's SaaS network monitoring providers, like Kaseya, LogicMonitor, Montis and Uptrends, can offer topology discovery, device monitoring, reporting, event collection and more. As you consider deploying cloud network monitoring, take into account the advantages and disadvantages that come along with using SaaS.

III. WINDOWS AZURE STORAGE (WAS)

Windows Azure Storage (WAS) is a scalable cloud storage system that has been in production since November 2008. It is used inside Microsoft for applications such as social networking search, serving video, music and game content, managing medical records, and more. In addition, there are thousands of customers outside Microsoft using WAS, and anyone can sign up over the Internet to use the system.

WAS provides cloud storage in the form of Blobs (user files),

Tables (structured storage), and Queues (message delivery).

These three data abstractions provide the overall storage and workflow for many applications. A common usage pattern we see is incoming and outgoing data being shipped via Blobs, Queues providing the overall workflow for processing the Blobs, and intermediate service state and final results being kept in Tables or Blobs.

An example of this pattern is an ingestion engine service built on Windows Azure to provide near real-time Facebook and Twitter search. This service is one part of a larger data processing pipeline that provides publically searchable content (via our search engine, Bing) within 15 seconds of a Facebook or Twitter user's posting or status update. Facebook and Twitter send the raw public content to WAS (e.g., user postings, user status updates, etc.) to be made publically searchable. This content is stored in WAS Blobs. The ingestion engine annotates this data with user auth, spam, and adult scores; content classification; and classification for language and named entities. In addition, the engine crawls and expands the links in the data. While processing, the ingestion engine accesses WAS Tables at high rates and stores the results back into Blobs. These Blobs are then folded into the Bing search engine to make the content publically searchable. The ingestion engine uses Queues to manage the flow of work, the indexing jobs, and the timing of folding the results into the search engine. As of this writing, the ingestion engine for Facebook and Twitter keeps around 350TB of data in WAS (before replication). In terms of transactions, the ingestion engine has a peak traffic load of around 40,000 transactions per second and does between two to three billion transactions per day

Strong Consistency – Many customers want strong consistency: especially enterprise customers moving their line of business applications to the cloud. They also want the ability to perform conditional reads, writes, and deletes for optimistic concurrency control on the strongly consistent data. For this, WAS provides three properties that the CAP theorem claims are difficult to achieve at the same time: strong consistency, high availability, and partition tolerance (see 8).

Global and Scalable Namespace/Storage – For ease of use,

WAS implements a global namespace that allows data to be stored and accessed in a consistent manner from any location in the world. Since a major goal of WAS is to enable storage of massive amounts of data, this global namespace must be able to address exabytes of data and beyond.

Disaster Recovery – WAS stores customer data across multiple data centers hundreds of miles apart from each other. This redundancy provides essential data recovery protection against disasters such as earthquakes, wild fires, tornados, nuclear reactor meltdown, etc.

Multi-tenancy and Cost of Storage – To reduce storage cost,

many customers are served from the same shared storage

infrastructure. WAS combines the workloads of many different customers with varying resource needs together so that significantly less storage needs to be provisioned at any one point in time than if those services were run on their own dedicated hardware.

Global Partitioned Namespace

A key goal of our storage system is to provide a single global

namespace that allows clients to address all of their storage in the cloud and scale to arbitrary amounts of storage needed over time. To provide this capability we leverage DNS as part of the storage namespace and break the storage namespace into three parts: an account name, a partition name, and an object name. As a result, all data is accessible via a URI of the form:

http(s)://**AccountName**.<service>1.core.windows.net/**PartitionName**/**ObjectName**

The **AccountName** is the customer selected account name for accessing storage and is part of the DNS host name. The AccountName DNS translation is used to locate the primary storage cluster and data center where the data is stored. This primary location is where all requests go to reach the data for that account. An application may use multiple AccountNames to store its data across different locations.

In conjunction with the AccountName, the **PartitionName** locates the data once a request reaches the storage cluster. The PartitionName is used to scale out access to the data across storage nodes based on traffic needs.

When a PartitionName holds many objects, the **ObjectName** identifies individual objects within that partition. The system supports atomic transactions across objects with the same PartitionName value. The ObjectName is optional since, for some types of data, the PartitionName uniquely identifies the object within the account.

This naming approach enables WAS to flexibly support its three data abstractions2. For Blobs, the full blob name is the PartitionName. For Tables, each entity (row) in the table has a primary key that consists of two properties: the PartitionName and the ObjectName. This distinction allows applications using Tables to group rows into the same partition to perform atomic transactions across them. For Queues, the queue name is the PartitionName and each message has an ObjectName to uniquely identify it within the queue.

VI. HIGH LEVEL ARCHITECTURE

Here we present a high level discussion of the WAS architecture and how it fits into the Windows Azure Cloud Platform.

Windows Azure Cloud Platform

The Windows Azure Cloud platform runs many cloud services across different data centers and different geographic regions.

The Windows Azure Fabric Controller is a resource provisioning and management layer that provides resource allocation, deployment/upgrade, and management for cloud services on the Windows Azure platform. WAS is one such service running on top of the Fabric Controller.

The Fabric Controller provides node management, network configuration, health monitoring, starting/stopping of service instances, and service deployment for the WAS system. In addition, WAS retrieves network topology information, physical layout of the clusters, and hardware configuration of the storage nodes from the Fabric Controller. WAS is responsible for managing the replication and data placement across the disks and load balancing the data and application traffic within the storage cluster.

WAS Architectural Components

An important feature of WAS is the ability to store and provide access to an immense amount of storage (exabytes and beyond). We currently have 70 petabytes of raw storage in production and are in the process of provisioning a few hundred more petabytes of raw storage based on customer demand for 2012.

The WAS production system consists of Storage Stamps and the Location Service (shown in Figure 1).

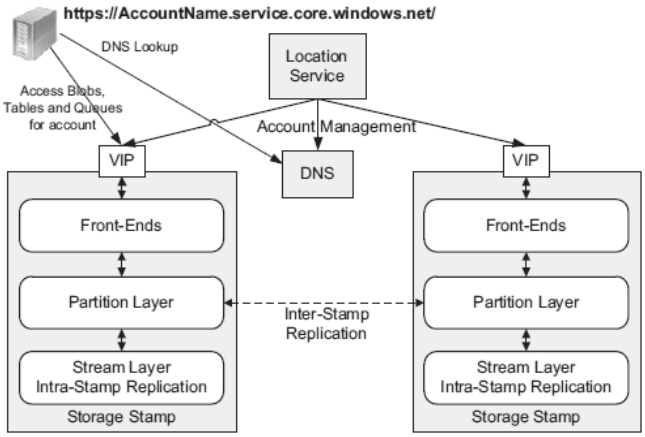


FIG. 3.1: HIGH LEVEL ARCHITECTURE

Storage Stamps – A storage stamp is a cluster of N racks of storage nodes, where each rack is built out as a separate fault domain with redundant networking and power. Clusters typicallyrange from 10 to 20 racks with 18 disk-heavy storage nodes per rack. Our first generation storage stamps hold approximately 2PB of raw storage each. Our next generation stamps hold up to 30PB of raw storage each.

To provide low cost cloud storage, we need to keep the storage provisioned in production as highly utilized as possible. Our goal is to keep a storage stamp around 70% utilized in terms of capacity, transactions, and bandwidth. We try to avoid going above 80% because we want to keep 20% in reserve for (a) disk short stroking to gain better seek time and higher throughput by utilizing the outer tracks of the disks and (b) to continue providing storage capacity and availability in the presence of a rack failure within a stamp. When a storage stamp reaches 70% utilization, the location service migrates accounts to different stamps using inter-stamp replication

Location Service (LS) – The location service manages all the storage stamps. It is also responsible for managing the account namespace across all stamps. The LS allocates accounts to storage stamps and manages them across the storage stamps for disaster recovery and load balancing. The location service itself is distributed across two geographic locations for its own disaster recovery.

WAS provides storage from multiple locations in each of the three geographic regions: North America, Europe, and Asia. Each location is a data center with one or more buildings in that location, and each location holds multiple storage stamps. To provision additional capacity, the LS has the ability to easily add new regions, new locations to a region, or new stamps to a location. Therefore, to increase the amount of storage, we deploy one or more storage stamps in the desired location's data center and add them to the LS. The LS can then allocate new storage accounts to those new stamps for customers as well as load balance (migrate) existing storage accounts from older stamps to the new stamps.

Figure 1 shows the location service with two storage stamps and the layers within the storage stamps. The LS tracks the resources used by each storage stamp in production across all locations. When an application requests a new account for storing data, it specifies the location affinity for the storage (e.g., US North).

The LS then chooses a storage stamp within that location as the primary stamp for the account using heuristics based on the load information across all stamps (which considers the fullness of the stamps and other metrics such as network and transaction utilization). The LS then stores the account metadata information in the chosen storage stamp, which tells the stamp to start taking traffic for the assigned account. The LS then updates DNS to allow requests to now route from the name https://AccountName.service.core.windows.net/ to that storage stamp's virtual IP (VIP, an IP address the storage stamp exposes for external traffic).

Three Layers within a Storage Stamp

Also shown in Figure 1 are the three layers within a storage stamp. From bottom up these are:

Stream Layer – This layer stores the bits on disk and is in charge of distributing and replicating the data across many servers to keep data durable within a storage stamp. The stream layer can be thought of as a distributed file system layer within a stamp. It understands files, called "streams" (which are ordered lists of large storage chunks called "extents"), how to store them, how to replicate them, etc., but it does not understand higher level object constructs or their semantics. The data is stored in the stream layer, but it is accessible from the partition layer. In fact, partition servers (daemon processes in the partition layer) and stream servers are co-located on each storage node in a stamp.

Partition Layer – The partition layer is built for (a) managing and understanding higher level data abstractions (Blob, Table, Queue), (b) providing a scalable object namespace, (c) providing transaction ordering and strong consistency for objects, (d) storing object data on top of the stream layer, and (e) caching object data to reduce disk I/O.

Another responsibility of this layer is to achieve scalability by partitioning all of the data objects within a stamp. As described earlier, all objects have a PartitionName; they are broken down into disjointed ranges based on the PartitionName values and served by different partition servers. This layer manages which partition server is serving what PartitionName ranges for Blobs, Tables, and Queues. In addition, it provides automatic load balancing of PartitionNames across the partition servers to meet the traffic needs of the objects.

Front-End (FE) layer – The Front-End (FE) layer consists of a set of stateless servers that take incoming requests. Upon receiving a request, an FE looks up the AccountName, authenticates and authorizes the request, then routes the request to a partition server in the partition layer (based on the PartitionName). The system maintains a Partition Map that keeps track of the PartitionName ranges and which partition server is serving which PartitionNames. The FE servers cache the Partition Map and use it to determine which partition server to forward each request to. The FE servers also stream large objects directly from the stream layer and cache frequently accessed data for efficiency.

Two Replication Engines

Before describing the stream and partition layers in detail, we first give a brief overview of the two replication engines in our system and their separate responsibilities.

Intra-Stamp Replication (stream layer) – This system provides *synchronous replication* and is focused on making sure all the data written into a stamp is kept durable within that stamp. It keeps enough replicas of the data across different nodes in different fault domains to keep data durable within the stamp in the face of disk, node, and rack failures. Intra-stamp replication is done completely by the stream layer and is on the critical path of the customer's write requests. Once a transaction has been replicated successfully with intra-stamp replication, success can be returned back to the customer.

Inter-Stamp Replication (partition layer) – This system provides *asynchronous replication* and is focused on replicating data across stamps. Inter-stamp replication is done in the background and is off the critical path of the customer's request.

This replication is at the object level, where either the whole object is replicated or recent delta changes are replicated for a given account. Inter-stamp replication is used for (a) keeping a copy of an account's data in two locations for disaster recovery and (b) migrating an account's data between stamps. Inter-stamp replication is configured for an account by the location service and performed by the partition layer. Inter-stamp replication is focused on replicating objects and the transactions applied to those objects, whereas intra-stamp replication is focused on replicating blocks of disk storage that are used to make up the objects.

We separated replication into intra-stamp and inter-stamp at these two different layers for the following reasons. Intra-stamp replication provides durability against hardware failures, which occur frequently in large scale systems, whereas inter-stamp replication provides geo-redundancy against geo-disasters, which are rare. It is crucial to provide intra-stamp replication with low latency, since that is on the critical path of user requests; whereas the focus of inter-stamp replication is optimal use of network bandwidth between stamps while achieving an acceptable level of replication delay. They are different problems addressed by the two replication schemes.

Another reason for creating these two separate replication layers is the namespace each of these two layers has to maintain. Performing intra-stamp replication at the stream layer allows the amount of information that needs to be maintained to be scoped by the size of a single storage stamp. This focus allows all of the meta-state for intra-stamp replication to be cached in memory for performance , enabling WAS to provide fast replication with strong consistency by quickly committing transactions within a single stamp for customer requests. In contrast, the partition layer combined with the location service controls and understands the global object namespace across stamps, allowing it to efficiently replicate and maintain object state across data centers.

V. AZURE SECURITY

An application's security is a function of its surface. The more surface that the application exposes the greater the security concerns. For example, an application that runs as an unattended batch process exposes less, from a security perspective, than a publically available website.

When you move to the cloud you gain a certain peace of mind about infrastructure and networking since these are managed in data centers with world-class security practices, tools, technology. On the other hand, the cloud intrinsically exposes more surface area for your application that can be potentially exploited by attackers. This is because many cloud technologies and services are exposed as end points vs. in-memory components. Azure storage, Service Bus, SQL Database (formerly SQL Azure), and many other services are accessible via their endpoints over the wire.

In cloud applications more responsibility lays on the shoulders of the application developers to design, develop, and maintain their cloud applications to high security standards to keep attackers at bay. Consider the following diagram (from J.D. Meier'sAzure Security Notes PDF): notice how the infrastructure part is being addressed by the cloud provider--in our case by Azure--leaving more security work to the application developers:

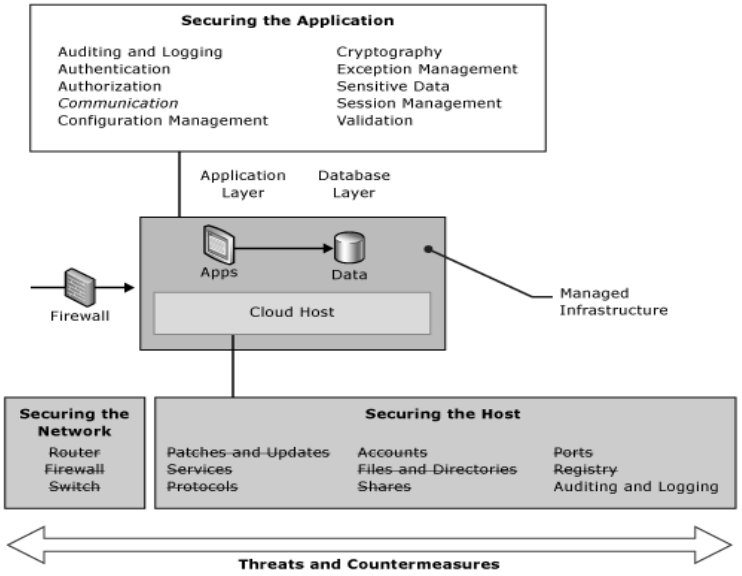


Fig. 4.1 Application security

The good news is that all of the security development practices, principles, and techniques you already know still apply when developing cloud applications. Consider the following key areas that must be addressed:

- All input is validated for type, length, range, and string patterns to avoid injection attacks, and all the data your app echoes back is properly sanitized.
- Sensitive data should be protected at rest and on the wire to mitigate information disclosure and data tampering threats.
- Auditing and logging must be properly implemented to mitigate non repudiation threats.
- Authentication and Authorization should be implemented using proven mechanisms offered by the platform to prevent identity spoofing and elevation of privileges threats.

Threats, Vulnerabilities, and Attacks

A threat is a potential bad outcome that you want to avoid such as the disclosure of sensitive information or a service becoming unavailable. It is common practice to classify threats by using the acronym "STRIDE":

1. **S**poofing or identity theft
2. **T**ampering with data
3. **R**epudiation of actions
4. **I**nformation disclosure
5. **D**enial of service
6. **E**levation of privileges

Vulnerabilities are bugs that we, as developers, introduce into the code that make an application exploitable by attackers. For example, sending sensitive data in clear text makes an information disclosure threat possible by a traffic sniffing attack.

Attacks are the exploitation of those vulnerabilities to cause harm to an application. For example, a Cross Site Scripting, or XSS, is an attack that exploits unsanitized output. Another example is eavesdropping on the wire to capture credentials sent in clear. These attacks can lead to an identity spoof threat being realized. To make it simple consider threats, vulnerabilities, and attacks as bad things. Consider the following diagrams as a balcony view of the bad things related to a Web application deployed to Azure

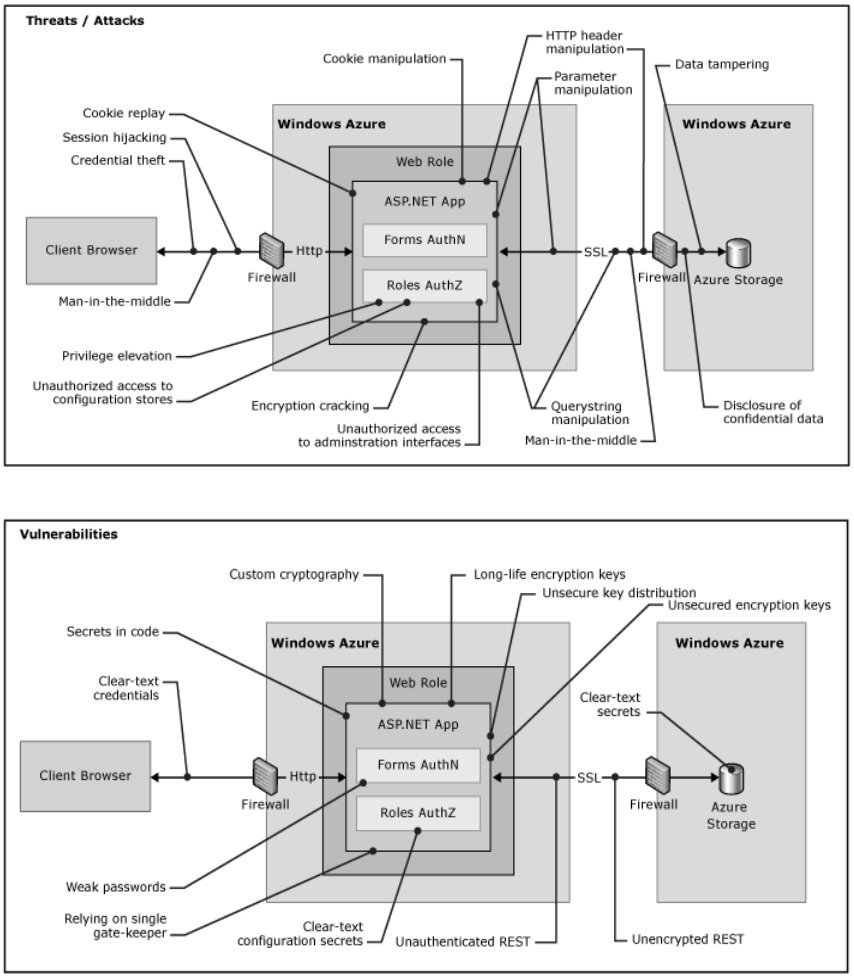


Fig. 4.2 Threats and Vulnarabilities

Identity and access related vulnerabilities leave you open to all threats in the STRIDE model. For example, an improperly implemented authentication mechanism may lead to an identity spoof and, as a result, information disclosure, data tampering, elevated privileges operations, or even shutting down the service altogether.

VI. AZURE IDENTITY AND ACCESS SCENARIOS

This section outlines common identity and access scenarios for different application architectures. Use the Scenarios Map for a quick orientation.

ASP.NET Web Form Application with Federated Authentication

In this application architecture scenario your web application may be deployed to Azure or on-premises. The application requires that its users will be authenticated by either corporate Active Directory or Internet identity providers.

To solve this scenarios use Azure AD Access control and Windows Identity Foundation.

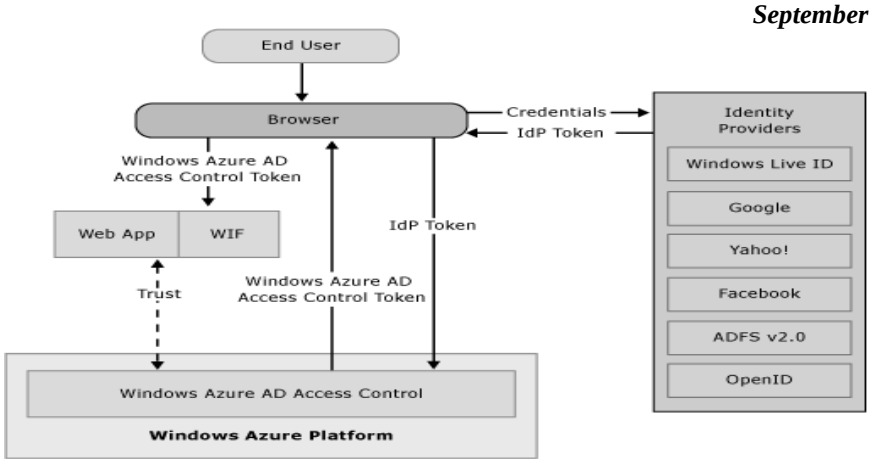


Fig. 5.1 Windows Azure Access control

WCF (SOAP) Service with Service Identity

In this application architecture scenario your WCF (SOAP) service may be deployed to Azure or on-premises. The service is being accessed as a downstream service by a web application or even by another web service. You need to control access to it using application specific identity. Think of it in terms of the type of app pool account that you used in IIS - although the technology is different, the approaches are similar in that the service is accessed using an application scope account vs. end user account.

Use the Service Identity feature in Azure AD Access control. This is similar to the IIS app pool account you were using when deploying your apps to Windows Server and IIS. Configure Azure AD Access Control to issue SAML tokens that will be handled by WIF at the WCF (SOAP) service.

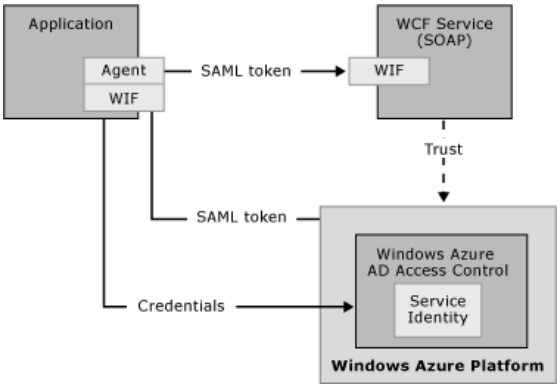


Fig. 5.2. WCF (SOAP) service

VII. CONCLUSION AND FUTURE WORK

This paper evaluates storage services provided by the leading cloud provider, WINDOWS AZURE. It shows how to create/own/access elastic compute cloud instance in all the possible ways. It deals with issues like bringing the machine up and down at anytime and creating users in a secure way with authenticated keys, which are very useful for an organization like universities, non-IT based companies that are willing to join AZURE Cloud. Finally, it provides some performance benchmark results to better understand the use of having instances on a cloud. One of the future works of this paper is to provide more information on better usage of the available resources on the cloud. Other works can be to incorporate all the services in different combinations (like EC2 with CloudFront, S3, etc.) and then benchmark the best performance, Create machine images from scratch, Security of data on the cloud, etc.

REFERENCES

[1] http://www.microsoft.com/windowsazure/.

[2] H.T. Kung and John T. Robinson, "On Optimistic Methods for Concurrency Control," ACM Transactions on Database Systems, vol. 6, no. 2, pp. 213-226, June 1981.

[3] J. Baker et al., "Megastore: Providing Scalable, Highly Available Storage for Interactive Services," in *Conf. on Innovative Data Systems Research*, 2011.

[4] F. Chang et al., "Bigtable: A Distributed Storage System for Structured Data," in *OSDI*, 2006.

