**Prototype for Cloud Storage Solution with Windows App:**

**Architecture Report & Findings**

Motifworks

**Table of Contents**

1 Introduction 3

2 Reference Architecture 3

3 Regions, Availability Zones (zone A and B here) and Edge Locations 7

4 Windows Based Cloud Application Prototype - Architecture 9

4.1 Aws Stack - 9

5 Technology Stack 11

6 Architectural Considerations 13

6.1 Authentication – 13

6.2 Switch AWS deployment regions – 14

6.3 Logging – 15

6.4 Windows App to Windows Desktop Application/Service communication – 15

6.5 Archival method – 15

6.6 Archive retrieval – 16

7 Cost Model 16

# Introduction

**Carl Zeiss India (Bangalore) Private Ltd** (**Zeiss**) is working on a prototype to build a cloud application that allows upload/download of images from/to windows tablets and storage of images and patients’ data on the cloud. This prototype involves development of a new application and storage infrastructure on Amazon cloud platform (AWS).

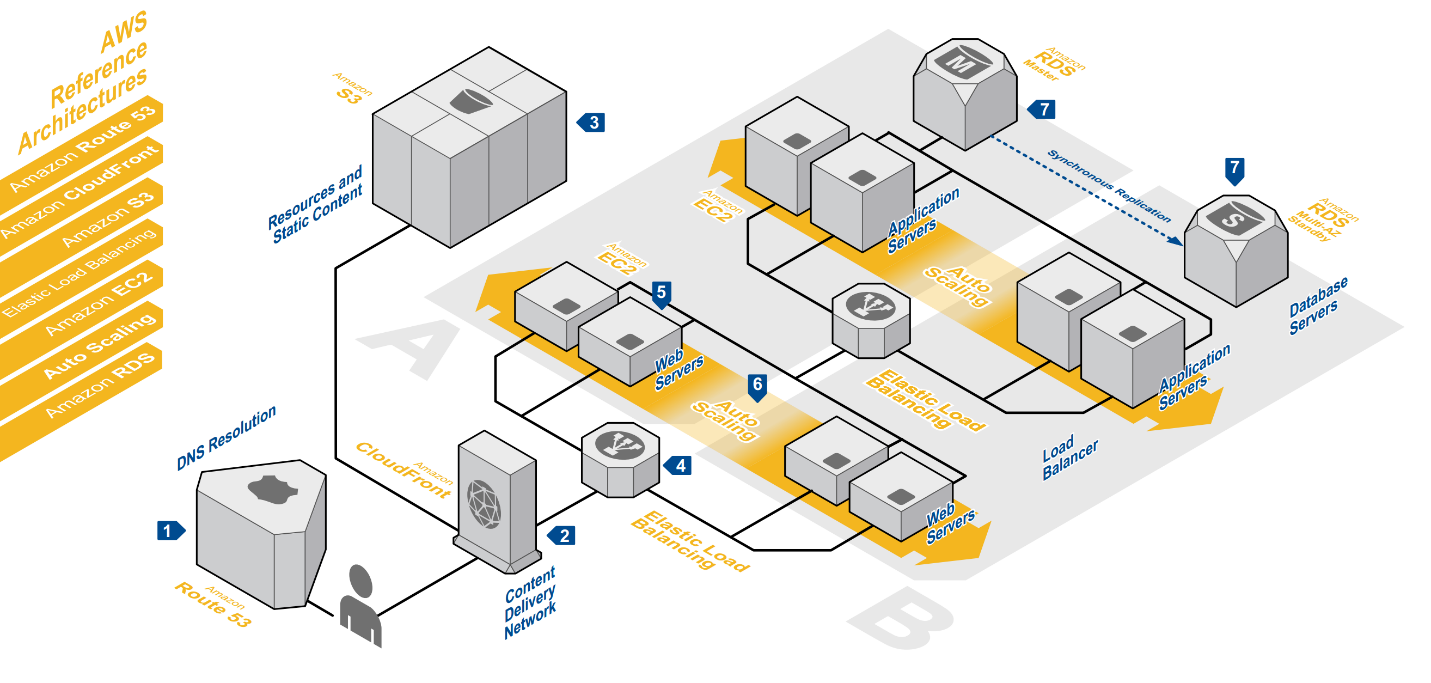
As one of its deliverables for this prototype, Motifworks has evaluated AWS and the available technology stack options for each component used in this prototype like RDS, Storage, Archival, Authentication etc. This document provides an evaluation framework for the actual product and suitability of Amazon cloud platform stack.

This document also covers architectural considerations like authentication mechanism, archive retrieval and storage, logging, deployment regions.

Finally, we have covered an indicative costing model for AWS based on our understanding of data and architectural requirements <To be added>

# Reference Architecture

Following architecture illustrates design of software application using services available on Amazon Web Services (AWS) cloud computing platform. This architecture allows to build a highly scalable, reliable, secure and high performance backend system to windows tablet/laptop application in a cost effective manner.



*AWS REFERENCE ARCHITECTURE for mobile-device/web app backend system*

1. **Amazon Route 52 for DNS Resolution –**

The user’s requests are served by Amazon Route 53, a highly available and scalable Domain Name System (DNS) web service. Route 53 effectively connects user requests to infrastructure running in AWS – such as Amazon EC2 instances, Elastic Load Balancers, or Amazon S3 buckets – and can also be used to route users to infrastructure outside of AWS.

AWS’s Route 52 is a preferred DNS service (as compared to numerous other DNS service providers) because of following reasons –

* Fast: Route 53 uses Anycast with 24+ DNS servers distributed across world. Queries to application domain name will be resolved by the nearest DNS server.
* Reliable: AWS guaranties 100% availability for Route 53.
* Integration with other services: It’s designed to work closely with other services like EC2, S3, CloudFront, etc.
* API Support: Ability to manipulate DNS is an important feature in Route 53. User can automate DNS management along with instance provisioning.

1. **Amazon CloudFront as Content Delivery Network –**

Amazon CloudFront is a content delivery web service. It integrates with other Amazon Web Services and provides an easy way to distribute content to end users with low latency and high data transfer speeds.

Static and dynamic content is delivered by a global network of edge location spread across the globe. Requests are automatically routed to the nearest edge location so content is delivered with the best possible performance.

For objects/static contents stored in AWS S3, either they can be directly distributed though Amazon S3 bucket or CloudFront service can be configured to get objects from S3 bucket and distribute objects from CloudFront.

If end- users access objects frequently, cost of delivering can be lowered by adding CloudFront because, at higher usage, the price for CloudFront data transfer is lower than the price for Amazon S3 data transfer. In addition, downloads are faster with CloudFront than with Amazon S3 alone because objects are stored closer to end-users.

1. **Amazon S3 for resources and static contents (patient images and reports) –**

Resources and static contents used by the system are stored on Amazon Simple Storage Service (S3). Amazon S3 provides a simple web-service interface to read, write and delete up to 5 terabytes of data each from anywhere on the web.

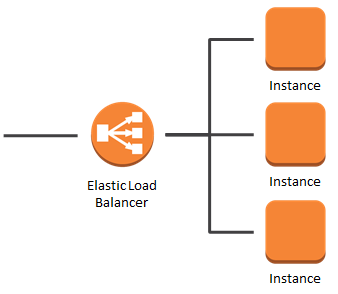
AWS S3, is a “highly durable and available store” and can be used to reliably store application content such as media files, static assets and user uploads. It allows user to offload entire storage infrastructure and offers better scalability, reliability, and speed than just storing files on any other file system. AWS S3 is very important when architecting applications for scale and is a perfect complement to Cloud based applications.

S3 is a different kind of file service and has different semantics from other file-based services. All files in S3 are stored in buckets which act as a top-level container much like a directory. All files sent to S3 belong to a bucket and bucket names must be unique across the whole Amazon system.

1. **Elastic Load Balancing (ELB) for automatic load balancing**

HTTP requests are first handled by Elastic Load Balancing which automatically distributes incoming application traffic among multiple Amazon Elastic Compute Cloud (EC2) instances across Availability Zones (AZs). It seamlessly provides load balancing capacity needed according to the traffic patterns allowing for efficient utilization of resources and greater fault tolerance.

It detects unhealthy instances and reroutes traffic to healthy instances until the unhealthy instances have been restored. ELB optimizes fault tolerance by adjusting capacity according to incoming application traffic. Users can enable ELB within a single availability zone or across multiple availability zones to maintain consistent application performance.



1. **Amazon Elastic Compute Cloud (EC2) hosted Web Servers and Application Servers**

Web servers and application servers are deployed on Amazon EC2 instances. Amazon Elastic Compute Cloud (Amazon EC2) provides resizable computing capacity in the Amazon Web Services (AWS) cloud.

Amazon EC2 provides virtual computing environments, known as *instances* with various configurations of CPU, memory, storage, and networking capacity for your instances, known as *instance types.*

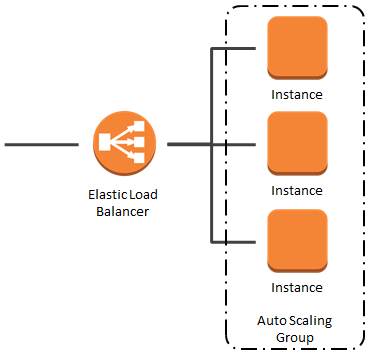
*From wikipedia –* ***Amazon Elastic Compute Cloud****(****EC2****) is a central part of amazon.com's cloud computing platform, Amazon Web Services (AWS). EC2 allows users to rent virtual computers on which to run their own computer application. EC2 allows scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image to create a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server instances as needed, paying by the hour for active servers, hence the term "elastic". EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy.*

*Amazon does not disclose information on internal working of Amazon AWS platform and its architecture.*

1. **Auto Scaling**

Web servers and application servers are deployed in auto-scaling groups. Auto-scaling automatically adjusts capacity up or down according to conditions defined/configured.

Auto scaling is used in conjunction with load balancing (ELB). When combined, the two features allow you to create a system that automatically adds and removes EC2 instances in response to changing load –



1. **Amazon Relational Database Service (RDS) for Relational Database Servers on cloud**

Amazon Relational Database Service (RDS) is a fully-managed SQL database service. Amazon RDS offers an array of database engine choices to help with database management tasks such as migration, backup, recovery and patching.

Amazon RDS provides access to MySQL, Oracle or Microsoft SQL Server database engines. Amazon RDS automatically patches the database software and backs up your database, storing the backups for a user-defined retention period and enabling point-in-time recovery. Compute resources or storage capacity associated with a Database Instance (DB Instance) can be scaled via a single API call.

In addition, Amazon RDS enables replication to enhance availability and reliability for production workloads. Mission critical workloads can be run with high availability (HA) and built-in automated failover from your primary database to a synchronously replicated secondary database in case of a failure.

# Regions, Availability Zones (zone A and B here) and Edge Locations

Amazon Web Services (AWS) currently operates in 8 regions globally. (*Note – there is an extra region called the AWS GovCloud region but this can only be used by government agencies*)

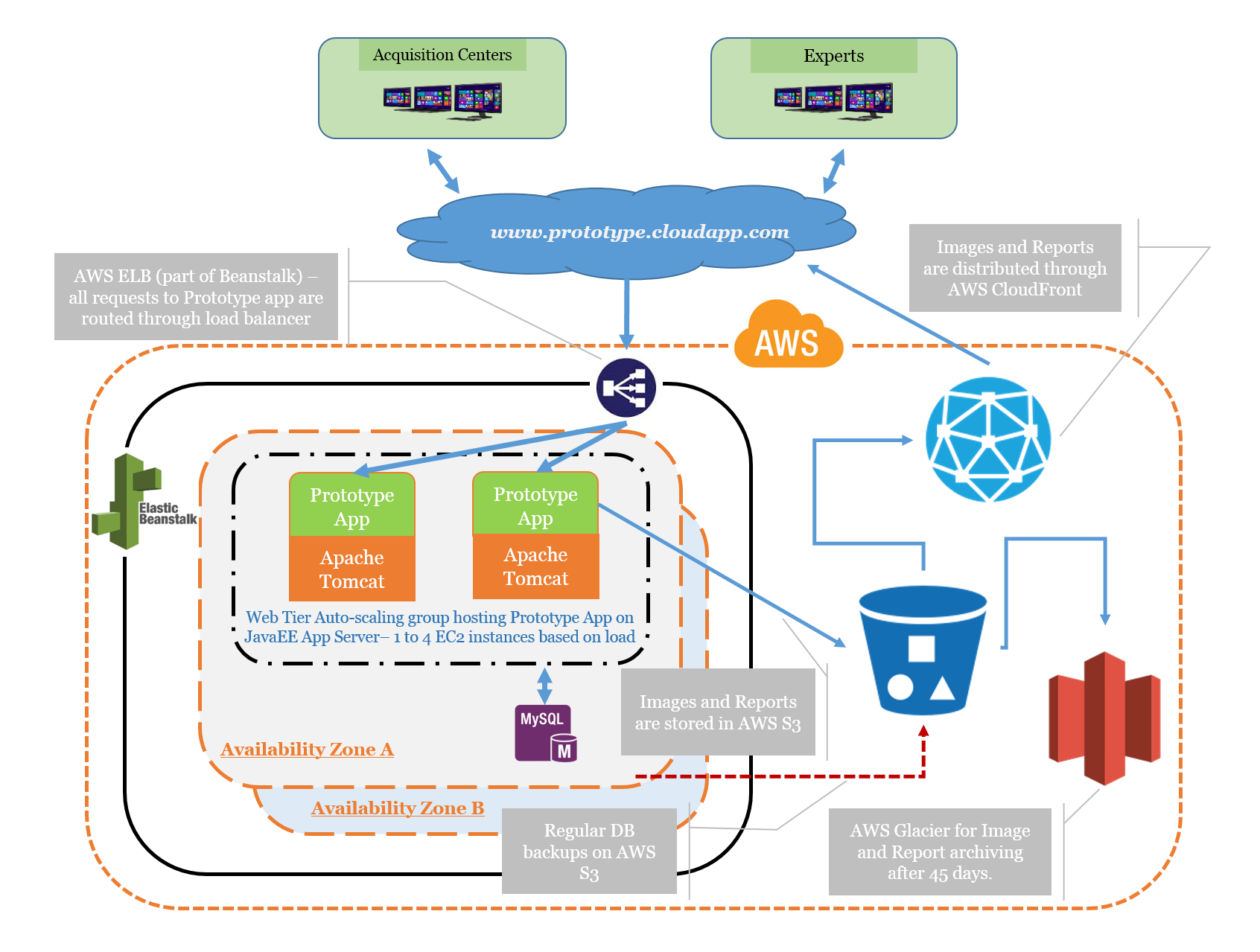


Although unusual, failures do happen. AWS therefore provides Availability Zones (AZ) that are isolated from each other (they have low latency connections). Deploying application across multiple availability zones makes it ready for unexpected outages. So if there is an outage with a particular availability zone, the application stays online.

Although user won’t have to worry about edge locations directly they are still an important part of the AWS infrastructure. There are currently 40 edge locations with more being added each month. They are located in most the major cities around the world and are used by CloudFront (CDN) to distribute content nearer to the end user.

# Windows Based Cloud Application Prototype - Architecture

Following diagram illustrates AWS cloud services based architecture for Windows based Cloud Application prototype -



## AWS Stack -

The stack is the core AWS Cloud container of AWS resources. Following AWS services will be used for cloud based architecture –

1. AWS Beanstalk –

AWS Elastic Beanstalk is a **PaaS (Platform as a Service)** service from Amazon Web Services that allows users to create applications and push them to a definable set of AWS services, including Amazon EC2, Amazon S3, Amazon Simple Notification Service (SNS), Amazon CloudWatch, auto scaling, and elastic load balancers. **AWS Beanstalk will be used for development and demonstration of Cloud App Prototype.**

Elastic Beanstalk is not a separate technology stack from AWS. In fact, it’s bringing together the traditional components from EC2, S3, RDS and CloudWatch into a fully scalable infrastructure.

* 1. **AWS Elastic Load Balancer (ELB) –**

All requests from windows based clients will be directed towards AWS ELB. ELB service will manage load balancing based on number of requests serviced by each EC2 instance.

* 1. **AWS Elastic Cloud Compute (EC2) –**

AWS EC2 will host Application and Web servers. For prototype, Cloud App will be developed using JavaEE technology platform and hosted on Tomcat Application/Web server platform provided by AWS Beanstalk.

* 1. **AWS Simple Storage Service (S3) –**

AWS S3 Buckets will be used to store patient images and reports. Initially 100 GB of data will be created on S3 bucket with 80% images of 2MB and 20% images of 5MB. Depending on analysis of 100GB data, up to 1 TB of data can be uploaded to evaluate performance and cost of AWS S3 service.

* 1. **AWS Relation Database Service (RDS) –**

AWS MySQL RDS will be used to Users, Patients, and Audit etc. data. Regular backups of Application data will be automated and stored on AWS S3 bucket.

* 1. **Auto-scaling –**

EC2 instances provided with Beanstalk are configured to scale automatically depending on server load. Default configuration of up to 4 instances will be utilized for prototype.

* 1. **Availability Zones –**

Beanstalk provides EC2 instances in 2 different availability zones to address fault tolerance.

1. AWS Glacier **–**

AWS Glacier is an extremely low-cost storage service that provides secure and durable storage for data archiving and backup. In order to keep costs low, Amazon Glacier is optimized for data that is infrequently accessed and for which retrieval times of several hours are suitable. With Amazon Glacier, customers can reliably store large or small amounts of data for as little as $0.01 per gigabyte per month, a significant savings compared to on-premises solutions.

AWS Glacier service will be used to archive images and reports that are older than 45 days. This storage service is one-third in price of any other storage service across cloud platforms and it will bring significant cost saving for storage of archive data (images and reports).

# Technology Stack

Elastic Beanstalk supports applications running on Java, Python, PHP and Ruby running on Linux, or IIS running on Windows Server. It also supports databases in MySQL, PostgreSQL, Oracle and Windows SQL Server.

1. **Tomcat/Java/Linux platform –**

Java web application technology stack combines capabilities of a true enterprise class technology like windows platform with flexibility of open source platforms.

Pros –

* Widely used in enterprise
* Professional support
* No cloud vendor locking - flexibility to choose or move cloud platform
* Multiple options for web and application servers to accommodate evolving requirements
* Excellent support of SDK and Libraries from Amazon
* Automation capabilities on Linux OS
* Excellent development tool/IDE

Cons –

* Not as lightweight as open source platforms like Ruby, Python or PHP

1. **IIS/Windows platform –**

IIS on Windows is widely used platform in enterprise solution space. It provides equal capability as Java platform for enterprise class applications.

Pros –

* Widely used in enterprise
* Professional support from Microsoft
* Excellent development tool/IDE

Cons –

* Larger footprints on cloud
* Higher cost with cloud infrastructure
* Lack of automation capabilities on Windows platform
* Azure is a better platform for Windows platform on cloud

1. **Python, PHP and Ruby with Linux platforms –**

Open source platforms are extensively used by SMBs on cloud platforms. These platforms provide cutting edge technology at lesser cost.

Pros –

* Cutting edge technology
* Faster development
* Lesser cost

Cons –

* No professional support
* Regular updates make it harder to maintain and migrate
* Security concerns due to open source nature

For prototype phase, Java technology stack with Tomcat web server deployed over Linux will be used. It will be easily possible to deploy application package (war file) over any other web or application server of choice in future. It also provides flexibility to choose any flavor of Linux/Unix OS without any changes to application code.

On Database layer, MySQL database will be used for prototype phase.

1. **MySQL –**

MySQL is most mature Relational Database Service (RDS) on Amazon AWS platform and most widely used database for web application. It lacks some of the features of most advanced databases like Oracle and MSSQL (e.g. limitation on triggers and joins) but it is equally capable for systems that are not highly transactional in nature (e.g. e-commerce site with millions of users)

Pros –

* Equally capable for scope of system under consideration
* Low cost
* Excellent MySQL RDS service provided by Amazon RDS

Cons –

* Limitation for highly transactional systems

1. **MSSQL and Oracle –**

Both MSSQL and Oracle are most widely RDBMSs in enterprise applications especially banking and e-commerce.

Pros –

* Widely used in enterprise
* Best in class

Cons –

* Expensive

1. **PostgreSQL –**

PostgreSQL is another RDBMS option available on Amazon RDS. PostgreSQL service is slightly expensive on Amazon AWS but it provides an equally capable option to MySQL.

1. **AWS DynamoDB (and other NoSQL DB like MongoDB) –**

DynamoDB is NoSQL database service provided by Amazon AWS. NoSQL databases provides benefits in terms of schema less database, simpler models, rapid development, distributed storage etc. but it lacks some of the basic features that are important for an enterprise applications – ACID, referential integrity, security, maturity etc.

Pros –

* Horizontal scalability
* Schema less with less joins
* Rapid Development
* Flexibility
* Easy to deploy Caching Layer
* Wide data types
* Less administration

Cons –

* ACID
* Maturity
* Analytics / BI / Reporting
* Security
* Data loss
* Remove duplicate data (Normalization)
* Referential integrity

# Architectural Considerations

## Authentication –

The following are some options that are commonly used for setting up authentication:

* **Basic authentication**: Basic authentication is one of the widely used ways to implement security as it does not require overhead of additional APIs, besides the APIs used in the implementation framework itself. As the name implies, it is basic in nature where you send user name and password in Base64 encoding with no advanced options. It should always be used with SSL encryption to prevent credentials being decoded.
* **OAuth 1.0a**: Oauth1 is a signature based protocol. It is a widely-used, well-tested, and very secure protocol. The protocol uses a cryptographic signature, such as HMAC-SHA1, which is a value that combines the token secret, nonce, and other request based information. The biggest advantage of OAuth 1 is that it does not require passing the token secret across the wire directly. Because of this, it completely eliminates the possibility of tapping the password from over the wire. This can be safely used without SSL. However, this higher level of security offered in this protocol comes with an overhead of complex signature generating process.
* **OAuth 2**: This protocol completely eliminates signatures; hence, it is much simpler. It requires Transport Layer Security, which handles encryption. There are not as many Oauth2 libraries as there are Oauth1a libraries, so integrating this protocol into API is more challenging. Because of this instability in the spec committee and because OAuth2’s default settings are less secure than OAuth1 (no digital signature means user can’t verify if contents have been tampered with before or after transit), OAuth1 is recommended over OAuth2 for sensitive data applications.
* **Custom security protocol**: This approach should be avoided not only because no one, besides developer of custom protocol, will be able to use custom protocol. It puts the responsibility of maintenance on the creator.

Because of advantages and disadvantages mentioned above, it is safer and simpler to choose Basic Auth with SSL for most REST services. However, high overhead based OAuth 1.0a protocol can be used for extremely sensitive data in less common situations.

**Microsoft Azure Active Directory –**

Azure Active Directory is not considered suitable for securing Cloud App APIs due to various reasons –

* Overkill - Directory service is not considered to be an optimal solution for securing RESTful APIs
* Integration with Amazon AWS hosted solution will have performance impact. It will require additional communication between AWS datacenter and Azure datacenter for every API request
* Lack of ease of user creation as compared to above mentioned options
* Higher cost as compared to above mentioned options

## Switch AWS deployment regions –

There is no simple process or tool from Amazon for migrating from one AWS region to another AWS region (there are paid third party tools available). Each AWS region is essentially treated as a separate cloud.   Migrating from one region to another is equivalent to migrating to a completely different datacenter.  Before making such a large scale change it's important that you take all things into consideration.

* Need to replicate EC2 image to the new region.
* Need to create the following resources in the new region: SSH Keys, Security Groups, Elastic IPs, S3 bucket.
* Need to update DNS settings accordingly.

## Logging –

Logs can be accessed from the Amazon EC2 instance running web or application server in 2 different ways –

* Elastic Beanstalk console
* Configure environment to automatically publish logs to an S3 bucket

User can configure your environment so that the logs from the Amazon EC2 instances running applications are copied by AWS Elastic Beanstalk to the Amazon S3 bucket associated with the application.

Logs are stored in folders that are organized based on the **beanstalk environment name** and the directory name would be the **ec2-instance's id** that generated the tomcat logs.

## Windows App to Windows Desktop Application/Service communication –

Windows 8 UI apps are intended to be sleek, reliable, fast, and touch-oriented. Window 8 UI apps are restricted in what parts of the file system, OS, and hardware they can access. Updating existing apps to this model introduces obstacles, with some app functionality being impossible to implement in the WinRT API. One solution is to create a Windows 8 UI front-end that communicates with a Window 8 Desktop app to perform the work not allowed by the WinRT API. There are a few ways to do this –

Web Service -

A Windows 8 Desktop app can be running as a backend with web service exposure, allowing the Windows 8 UI app to connect and communicate. The Desktop app can be also used as a mediator, handling the interactions and receiving requests over the connection.

Local Files (recommended approach) -

When the two apps are on the same machine and are attempting to communicate without using the network, the limited options make things difficult but not impossible. The apps could communicate via local files they can both access. Windows App can modify the shared files to enable communication in almost real time by writing and reading the files quickly.

Apps can access certain file system locations by default. Apps can also access additional locations through the file picker, or by declaring capabilities. App installation folder, app data folder, download folder are some of the folders accessible to an app by default.

In addition to the default locations, an app can access additional files and folders by declaring capabilities in the app manifest, or by calling a file picker to let the user pick files and folders for the app to access.

Cloud - Storage and Notifications

If network access is available, both of the apps can share files on a remote server or data cloud. Most cloud services have safeguards in place to prevent file access collisions and data loss. Setting up a Windows Push Notification Server (WNS) allows the Desktop app to send messages and updates as notifications to the Windows 8 UI app. The notifications can be “toast” style, live tile updates, or handled by code for customized communication.

## Archival method –

Combination of AWS S3 and AWS Glacier service will be used for archival of image data. Lifecycle rules are setup using following elements on S3 objects –

* A prefix to specify which objects in the bucket are subject to the policy.
* A relative or absolute time specifier and a time period for transitioning objects to Glacier. The time periods are interpreted with respect to the object's creation date. They can be relative (migrate items that are older than a certain number of days) or absolute (migrate items on a specific date)
* An object age at which the object will be deleted from S3. This is measured from the original PUT of the object into the service, and the clock is not reset by a transition to Glacier.

User can create a lifecycle rule in the AWS Management Console. Every day, S3 will evaluate the lifecycle policies for each of your buckets and will archive objects in Glacier as appropriate. After the object has been successfully archived using the Glacier storage option, the object's data will be removed from S3 but its index entry will remain as-is. The S3 storage class of an object that has been archived in Glacier will be set to GLACIER.

## Archive retrieval –

User can use S3's new RESTORE operation to access an object archived in Glacier. As part of the request, user need to specify a retention period in days. Restoring an object will generally take 3 to 5 hours. Restored object will remain in both Glacier and S3's Reduced Redundancy Storage (RRS) for the duration of the retention period. At the end of the retention period the object's data will be removed from S3; the object will remain in Glacier.

Although the objects are archived in Glacier, user can't get to them via the Glacier APIs. Objects stored directly in Amazon Glacier using the Amazon Glacier API cannot be listed in real-time, and have a system-generated identifier rather than a user-defined name. Because Amazon S3 maintains the mapping between your user-defined object name and the Amazon Glacier system-defined identifier, Amazon S3 objects that are stored using the Amazon Glacier option are only accessible through the Amazon S3 API or the Amazon S3 Management Console.

# Cost Model

*<TBD>*