Migration Scenario: Migrating Backend Processing Pipeline to the AWS Cloud

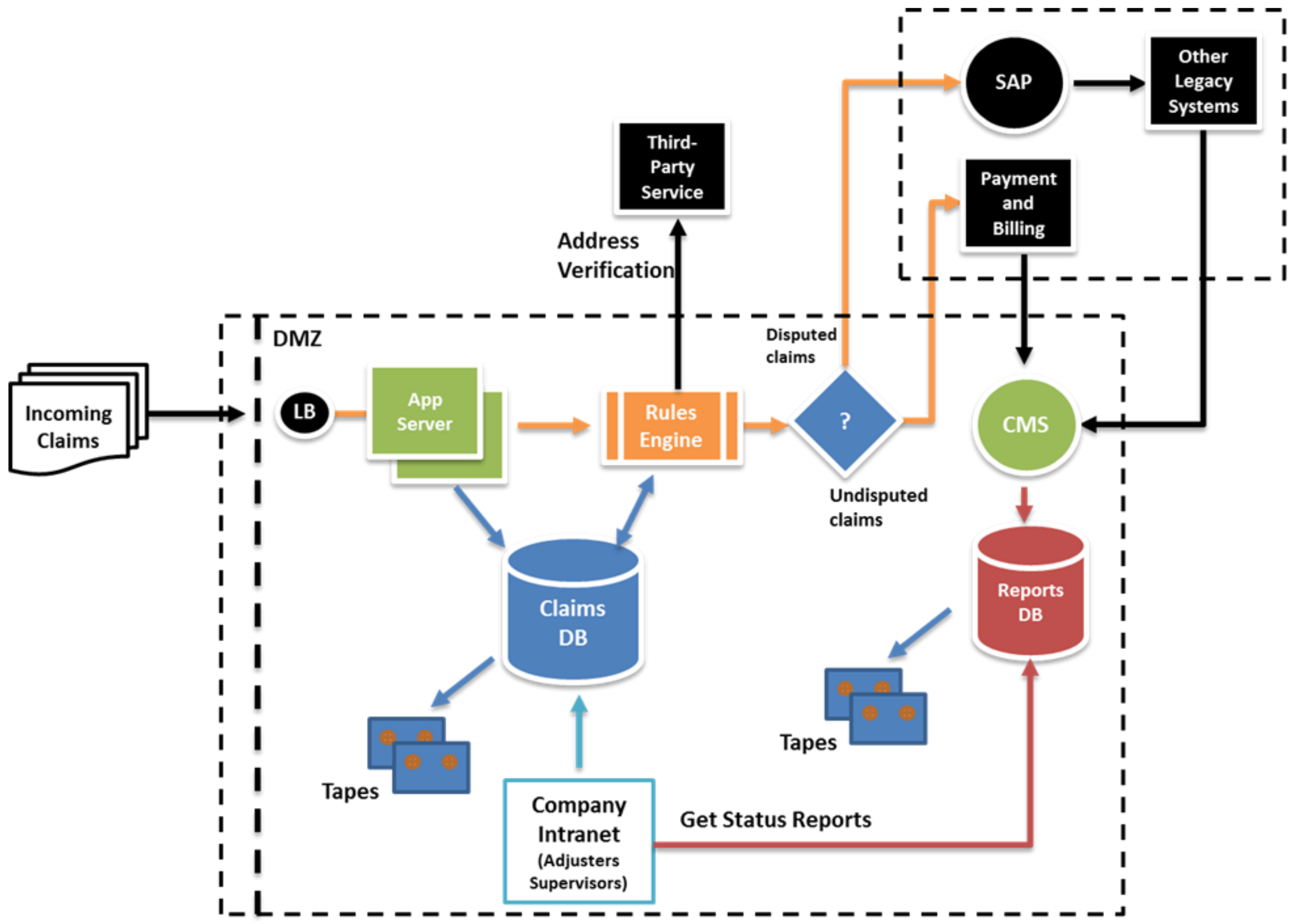

Figure 1: Company C Architecture (Before Migration)

Use case

Company C is an automobile insurance claim processing company with an external-facing website from which claims are received and processed. A backend processing pipeline processes the claims by connecting to a complex workflow of third party services, legacy systems, content management systems (CMS), and reporting databases. The company maintains an internal back-office portal (Intranet) for Adjusters and Supervisors to manage the end-to-end workflow of processing claims.

Application Architecture

Incoming claims from the company's website and various desktop clients are processed by the backend processing pipeline. See Figure 1 for an illustration of the current architecture. This pipeline consists of application servers that ingest the claims into a normalized relational database. Claims are then validated using the Rules Engine. The Rules Engine verifies and validates the data in the claims with the help from internal and external services like the 3rd party

Address Verification Service. Both the application server and Rules Engines are built using standard .NET technology. All the validated data is again stored in the same database which is backed up to tape drives periodically. Disputed claims undergo a complex workflow process and are processed using legacy systems. Undisputed claims are directly fed to the payments and billing system. The entire claims processing workflow is audited and logged in Reports DB running on MySQL DB, and central Claims DB using a SQL Server database. The company Intranet which is mainly used by Supervisors and Adjusters tracks all the claims and manages the entire process. Company Intranet is a web-application that provides basic user management and authentication. An open source Content Management System (CMS), manages evidence and supporting documents along with claims documents and reports.

Motivation for Migration

Company C would like to reduce its IT infrastructure investment and cut costs by lowering the total cost of ownership (TCO), and reducing the overall spend on IT administration and storage. Company C has a steady demand for claims processing throughout most of the year. Occasionally, the company gets a surge of claims that needs to be processed, and this puts strain on the current infrastructure. Hence, the company would like to scale on-demand servers without having to provision for peak capacity due to the sunk costs associated with purchasing additional servers upfront.

Last year, Company C was affected by an extended period of downtime due to system failure that resulted in loss of revenue and a sub-optimal customer experience. Company C would like to implement a low-cost highly reliable Disaster Recovery Solution for its mission-critical claim processing infrastructure so that in an event of another system failure, power outage, accident or a natural disaster, the company can failover to a different site in order to get around-the-clock business continuity. The company was slightly skeptical of moving their mission critical claims processing system all at once into the cloud, and decided to first leverage the business continuity benefits of the cloud, and then migrate the entire infrastructure into the cloud.

Migration Plan, Strategy, & Execution Steps

Cloud Assessment

As a first step, the company conducted a business assessment of the cloud, and analyzed all licensed products. They found out that AWS has teamed with most of the software and technology vendors who are offering equivalent EC2-based “pay-as-you-go” licensing. During the financial assessment, the company also calculated they would save hundreds of dollars every month in storage costs by just moving the data from tape drives to Amazon S3 for data backup purposes. The company will be able to use most of the internal system management tools without any major modifications. They learned that the cost of implementing and maintaining a parallel infrastructure to achieve business continuity will be quite less as the “hot site” simply consists of an array of EC2 instances with mounted EBS volumes and a set of pre-configured AMIs.

During the technical assessment, the development team classified all the major logical components of the claims processing system and created a dependency tree spreadsheet that listed all the applications with upward and downstream dependencies to other components and applications. The team also recognized that the CMS and Reports system had the fewest dependencies, and could be moved to the Cloud first. For business continuity, the team would have to configure the AMIs and EBS volumes. The AMIs would have to be updated every six months with new software releases, while the data in the two databases needed to be replicated every hour to the cloud.

Proof of Concept

To validate the technology and familiarize themselves with the AWS platform, the team decided to move the application server, database, of CMS system to the cloud. They moved the Reports DB and file repository first to Amazon RDS (since it was based on a MySQL database) and Amazon S3 respectively. This was followed by the moving the CMS, which was deployed to a fleet of EC2 instances. The team performed a variety of functional, stress and verification tests.

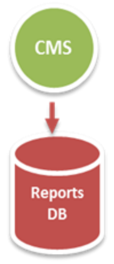

Figure 2: CMS System

When moving the reports database, they team used Amazon RDS, and using the standard MySQL import tools (*mysqldump*), they instantiated RDS instances, and imported the respective database dump file. Documents and files of the CMS were moved to Amazon S3 and reference paths (URLs) were changed to point to S3 objects.

After the proof of concept, the team gained more confidence in AWS and realized that they could move more primary components of the claims processing system to AWS instead of just using the cloud for storage and disaster recovery.

Data Migration

All the old archived data on tape drives were copied to hard drives and shipped to AWS using the Amazon Import/Export Service, which uploaded all data to designated Amazon S3 buckets. Backup of all the new data was handled by a PowerShell script that invoked a scheduled task performing the full database backup to Amazon S3 using command line tools (See Figure 2). In moving the central claims database, the team used standard SQL Server migration tools to move the databases to SQL Server based Amazon EC2 instances. Additional backups were performed using EBS snapshots.

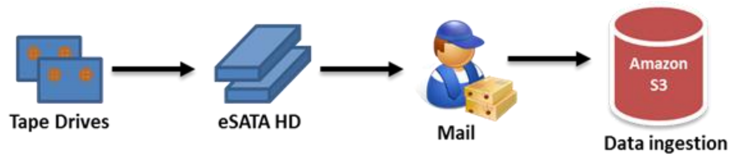

Figure 3: Data Migration from Tape to Amazon S3 using Amazon Import/Export Service

Application Migration

The Data Migration and Application Migration phases were executed in parallel. The goal of the application migration phase was to create an identical setup (clone) of the claims processing system in the cloud such that it could be brought to life within minutes.

Using the forklift migration strategy, the team was able to move most of the components – App Server and Rules Engine application on Windows server instances in EC2. Scripts run periodically and take snapshots of EBS volumes every day. AMIs of every instance type were created and were configured such that instances can be stopped and restarted easily. Data was replicated from the Claims DB every hour to the cloud using bi-directional transactional replication. The team created additional scripts to monitor the health of the system and application availability, and to notify on-call system administrators in case of any suspicious behavior in order to bring up the entire application in the cloud within a few minutes (cloud-based disaster recovery solution).

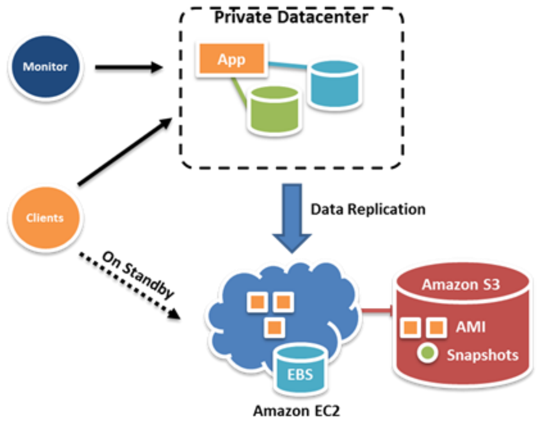

Figure 4: Disaster recovery in the cloud

To integrate applications running in the cloud with legacy applications running within the existing datacenter, the team evaluated two approaches:

1. Create a VPN tunnel between legacy and cloud applications (using Amazon VPC)
2. Write Web services wrappers around legacy applications and expose them as Web Services and glue the components with queues to make them "cloud aware"

Since the web ports were accessible outside of the enterprise network, the team implemented solution #2. Most of the intersection components were replaced by Amazon SQS queues to create a buffer and isolation between the various components of the processing pipeline. The proxy "cloud aware" daemons poll the queue for new messages, and as a result make the system more loosely coupled and hence highly scalable.

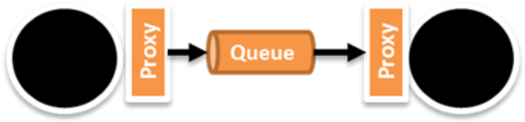

Figure 5: Expose legacy systems as web services

Leveraging the Cloud

The company realized that now that they have built a cloud-based architecture that can be brought up to life within minutes, they now implemented an HA-proxy based load balancing solution that routed user traffic above a certain threshold of 300 claims or from certain parties to the AWS cloud. This solution not only helped them to utilize the existing hardware as much as possible, but also switch back and forth between the old, on-premise and the new, cloud based architecture.

The Rules Engine application was re-packaged and re-engineered to run standalone on one EC2 instance. In order to make it elastic, a pre-configured AMI containing the binaries were created so that it can be spawned when needed. After some testing, multiple instances of the rules engine could run and would process the jobs faster than before. In order to tighten the security measures, the team implemented the security best practices at every stage, for example, only rules engines can update the data in Claims DB, only authorized developers were allowed to spawn the Rule Engine AMI with keys rotated every month, only certain security groups were used etc. The benefits of auto scaling and elasticity encouraged the team to run production rule engine instances in the cloud.

The company also realized that they have the entire automated infrastructure to achieve higher availability by adding more redundant infrastructure across availability zones (AZs). With just few modifications, they could bring the entire system up in a different AZ.

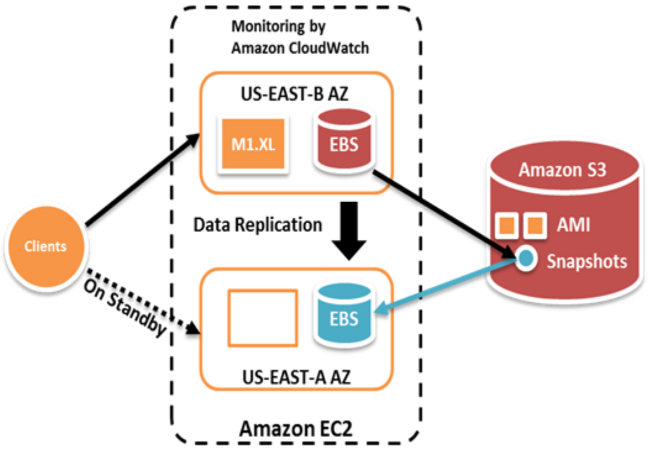

Figure 6: High Availability in the cloud

Optimization

After moving components into the cloud, the development team was able to make suggestions and recommendations to the senior management on how they could efficiently manage the capacity, and further reduce the overall spending in IT infrastructure. For example, running the rules engines only when it was needed and turn it off when it was not needed (during weekends) and save cost.

The development team had an opportunity to tune the software to run more effectively on cloud-based virtual hardware. They were able to implement caching and even further reduce their IT footprint and immediately see cost savings reflected in the next month's bill.

Conclusion

Company C was able to successfully migrate an existing backend processing system in the cloud. They were initially looking to use the cloud as their Disaster Recovery site. However, the benefits of the elasticity and reliability of the cloud encouraged them to run their primary components in the cloud. The final architecture is shown in Figure 7.

The company was able to easily map components from the on-premise architecture to the cloud-based architecture. Using the cloud, the company immediately gets the benefit of scale at all components of the architecture with the added benefits of lower cost and application deployment agility.

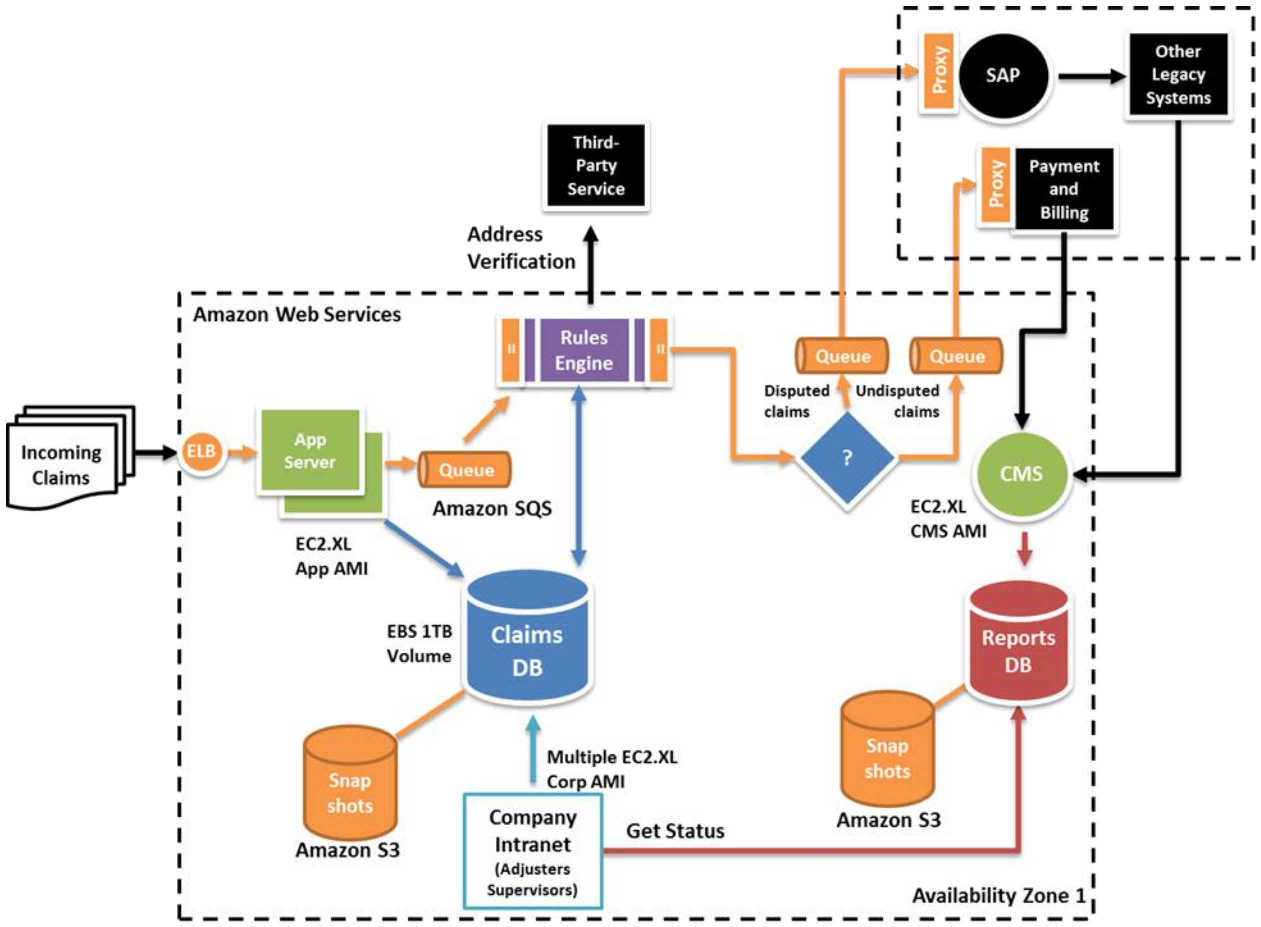

Figure 7: Company C architecture (After Migration)