Capstone Proposal Project Name: **Enterprise SaaS Migration and Modernization**

Student Name: **Timothy James Clark**

Degree Program: **BS Information Technology – Software Development Emphasis**

Mentor Name: **Paul Brown**

Signature Block:

Student's Signature: Timothy J. Clark

Mentor's Signature: _____

Table of Contents

Capstone Introduction

As of 2017 it was reported that the world's most valuable resource is no longer oil, rather data. While this is most clear in technology giants such as Amazon, Google, and Facebook the underlying truth remains throughout all companies large or small, regardless of the industry which they may operate within. In today's world when we hear the word data, we think of massive datasets that reside within servers and databases around the globe. However, there is still an abundance of companies ranging from small startups to large corporations in non-technology focused industries which have continued utilizing out of date methods of data collection and data storage amassing large quantities of physical datasets. Businesses such as law firms are a wonderful example of this physical data phenomenon. For example, one legal case may have up to 10,000 pages to a formal document. With an average file cabinet holding only up to 2,500 legal format documents this means each individual formal case could require as many as four physical file cabinets. With the majority of legal cases requiring a five-year retention policy on all legal documents, this means a massive amount of physical data storage can be required for even a small law firm. Not only does this increase the physical space required for the law firm to retain the physical data however it also poses a very large risk in the event of a natural disaster, flood, or fire where the documents which they are legally obligated to retain may be destroyed.

XYZ Incorporated is a small software development company that provides an enterprise-ready solution to the issue of physical data collection and storage through a Software as a Service (SaaS) document management solution. Their headquarters is in Huntsville, Alabama with a large majority of the employees working from their own homes throughout the United States. XYZ has been relatively successful in the document management sector since its initial launch in 1999. The first release of the companies document management system came on Microsoft's Windows 95 operating system and included a software offering that ran on a single individual PC acting as both the client as well as the server. Soon after the initial release

on Windows 95 came an upgraded release onto Windows 2000 followed shortly by another upgraded release onto Windows XP. However, throughout this period the software's design remained the same in which a single PC would act as both the client as well as the server. Following the multiple platform releases, XYZ decided it was time to utilize the development teams expanded knowledge to make a change that would change the product and company's future forever. XYZ's development team spent the next few years working on a new version of the document management system which would now allow for a true client-server environment where multiple PCs could connect to a single backend server. This new version required multiple changes to the document management system's back-end which led the development team to a design which included a new Microsoft SQL Server database to store all metadata, repository file folder for the physical files, and a windows service running the integral back-end server services all located on the new back-end server. While the PC users would simply need a new thin client with a one-time configuration at setup to point the clients to the newly installed back-end server's IP address. Customers had the option to purchase either individual licenses for each of their users or shared licenses which could be placed into a pool on a first come first serve basis for users to share at a slightly higher cost per license.

Over the following years, XYZ's sales team began to hear a reoccurring theme as it was still early in the 2000's many businesses did not have available resources either in hardware nor staff to successfully manage the client-server application on their own. Though XYZ had an immaculate technical support team available to all customers as part of their reoccurring maintenance and support fee model, many customer executive teams still lacked the confidence in their internal resources to move ahead with the adoption of the software. XYZ's support team echoed these concerns through historical support ticket data noting a very large trend in supporting the back-end server over the PC thin client. XYZ's CEO decided to make an at the time unprecedented decision to offer a hosted version of the document management solution to customers. This decision was met with strong hesitation by the development team as well as the

other members of the XYZ executive team. As at that time, it was 2005 and the cloud as we know it today was still vastly unexplored and unproven with offerings such as AWS, Azure, and GCP not existing to public knowledge.

Following months of software revisions to both the PC thin client as well as the server back-end service, the development team was finally able to reach a customized deployment of the document management system. This version of the software included a SQL Server back-end which held side by side customer databases within a single SQL Server Instance. The SQL Database schema had also been overhauled through this process to allow settings to now be stored within the database itself including usernames and passwords as well as most importantly a specified path for the customer's specific file repository path allowing for the separation of physical files based on the customer. However, the issue of how customers would connect back to this now XYZ hosted version of the document management system remained. XYZ's CEO and development team landed on what would now be considered a somewhat complicated and cumbersome solution to a then rather unexplored problem. The solution was to host the server within the XYZ office on a private server and having the end-users connect via VPN prior to logging in to their thin client on their PC.

Over the following five years this is the basis of how the hosted version of the document management system existed and operated. However, in April 2011 a string of tornados struck Huntsville, Alabama which took out the power for the city for nearly an entire week. This unfortunate natural disaster led to an unforeseen eye-opening experience for XYZ as its enterprise customers were unable to access their business-critical data while XYZ's office remained without power. Though the XYZ team thought quickly and worked together to obtain gas-powered electricity generators which were utilized to power the critical servers utilized to run the back-end services for the VPN and document management system, it was not before much of the customer trust had been lost and XYZ was put on the ropes to act fast in executing a plan to ensure an issue like this could not occur again in the future. The solution which was

decided upon by XYZ's CEO was to move the physically hosted server into a private datacenter in Nashville, Tennessee. However, in addition, the development team was also to implement a back-up routine that would replicate a bit-by-bit copy of the server across the internet to another private data center located in Birmingham, Alabama. In theory, this is a very practical and typical disaster recovery solution utilized by many companies even to the current day. However, this was not an extremely practical solution in terms of infrastructure management now burdening the development team nor was the project executed in a well thought out process rather in a haste following the outage in an attempt to restore customer faith.

Quickly after the initial migration occurred both the software development team, as well as the XYZ executive team, realized it was time to become more serious in their efforts of managing their hosted offering which now accounted for nearly 60 percent of the company's annual revenue. This decision led to my hiring as the Principal Cloud Architect of XYZ. Upon beginning with XYZ I was tasked with one main item which was the day to day operations management of the hosted version of the document management system. Following an extensive initial review of the hosted infrastructure, it was clear the current configuration in place needed an overhaul and thus I presented to the XYZ executive team my plans for a long-term project focused on the migration and modernization of the hosted document management system. The project recommended at least three primary project phases:

1. Establishing a suitable cloud hosting infrastructure for the application server.
2. Performing a stable and complete migration to the new environment.
3. Modernizing the server infrastructure for increased performance and scalability.

As part of the preparation for this project, in-depth analysis and evaluation of the existing architecture and all dependencies were performed. The outcome of the analysis would then be utilized in concluding which areas offered the most room for improvement. During this review, I was able to identify an existing server infrastructure which is pictured below.

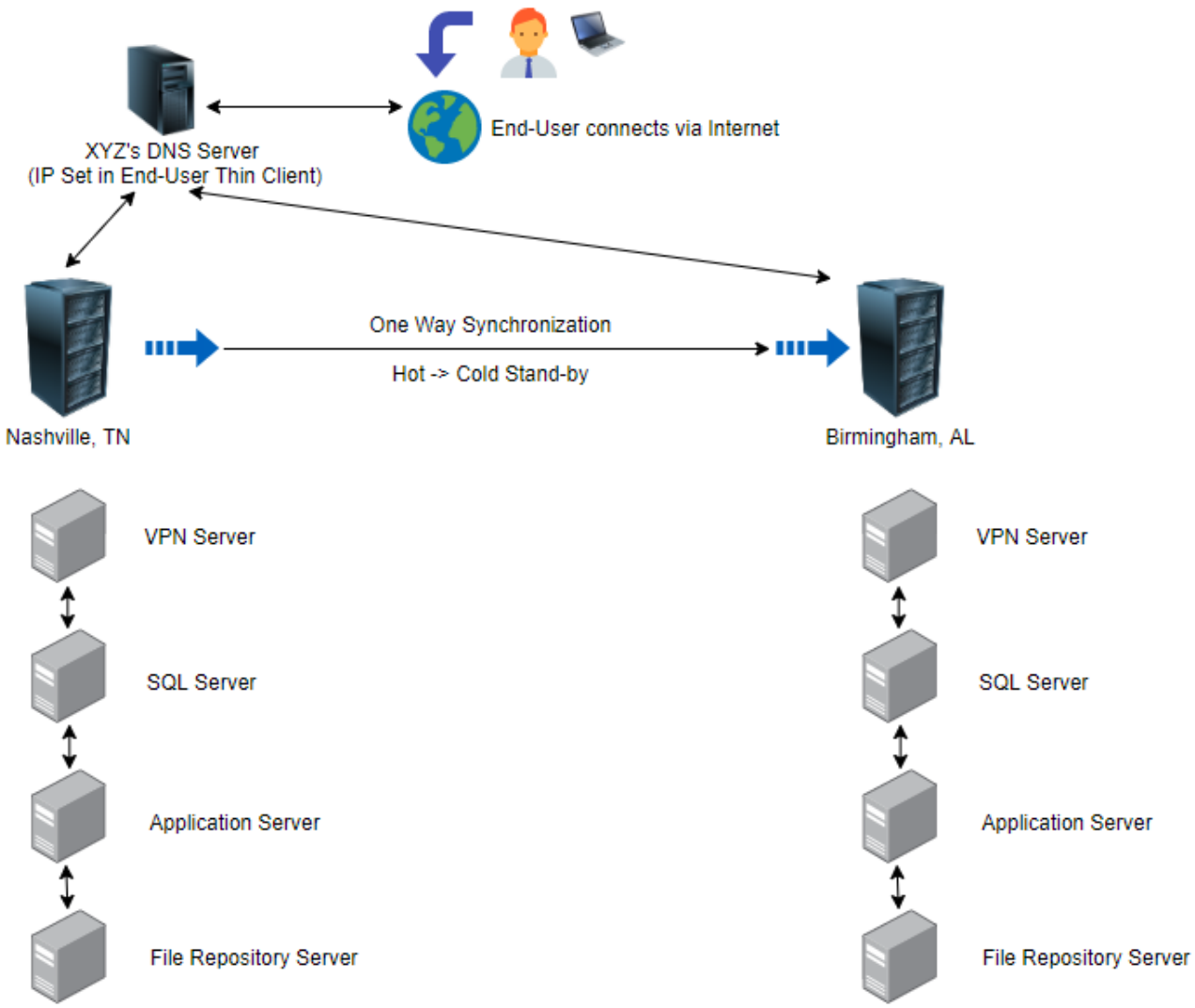


Figure 1. Diagram depiction of server infrastructure in place prior to project initiation.

Once the initial infrastructure review had been completed, I was able to bring my suggestions as the subject matter expert on the cloud to the key stakeholders within XYZ, including the CEO, CTO, development lead, and senior production engineers. As much of the future revenue and continued rapid growth of the company's sales depended immensely on the hosted version of the application the leadership and development teams would need to play a critical role in ensuring I had a full understanding of the requirements and available development resources if any code changes were needed to handle any underlying infrastructure changes which may require adjustments to be made. The consensus among all stakeholders involved was the need for a more reliable and scalable hosted infrastructure that in the event of

a major disaster would have a higher level of fault tolerance built in to allow for a seamless transition remaining in an always-on state. Additionally, they explained reports had begun to come in following the previous server migration of decreased speed and performance of the application from end-users. This meant that it was also very important that the application was modernized, and performance was increased wherever it seemed possible during the platform overhaul.

By and large, the main objective of this project was to deliver an improved application server infrastructure in a solution that surpasses management's expectations while remaining within budget. At the conclusion of the project, I will provide the stakeholders a cost-effective, scalable, fault-tolerant, and high-performing server infrastructure that will not only provide an improved experience for both XYZ and its customers but also a future proof framework to support future company expansion within the cloud. The design proposal for this project will highlight the Systems Development Life Cycle (SDLC) methodology. This style of process management is extremely common in the software industry as it allows for the planning, analysis, design, and implementation of an application infrastructure followed by a maintenance phase. If adjustments are found to be needed to the infrastructure based on application utilization growth or underlying application phases it can then leave the maintenance phase and recycle through the development cycle.

Review of Other Work

Because of the company's deep integration as a Microsoft partner and the development team's focus on the .NET framework with a Microsoft SQL database back-end, Microsoft's Azure cloud platform was the preliminary proposal for the next platform migration destination. However, since management sought to have additional alternatives available, Amazon's AWS platform was additionally assessed. After an extensive review, the findings were impressive for both platforms with AWS holding a larger market share with a more expansive set of products

currently available. However, Microsoft's Azure platform had the promise of an equally reliable, scalable, and fault-tolerant solution while coming in at a lower-price do to XYZ's partnership with Microsoft and the inclusion of software licenses used for the application's back-end such as Microsoft SQL Server and Microsoft Windows.

Azure Hybrid Benefit for SQL Server is an Azure-based benefit that enables customers to use SQL Server licenses with Software Assurance or qualifying subscription licenses to pay a reduced rate (base rate) on SQL Database vCore-based options, such as Managed Instance, vCore-based Single Database, and vCore-based Elastic Pool; on SQL Server in Azure Virtual Machines (including, but not limited to, Azure Dedicated Host); and on SQL Server Integration Services. (Microsoft Corporation, 2019)

As staying within budget was only one important factor of the project it was equally as necessary to ensure the migration was able to be completed smoothly to avoid disruption to XYZ's customers. To address concerns of a difficult migration an in-depth review of Microsoft's Azure white papers and customer stories was performed. Upon initial review, I came across a story focused on one of Microsoft's professional service partners iManage. iManage had recently completed a migration of their own customer-facing SaaS application in a straight forward rehosting or otherwise called lift and shift, in which the application is migrated in its current state from an existing private cloud into the public cloud such as Azure in this instance.

Through its market-leading software, iManage helps more than 2,000 of the world's largest law firms and 500 corporate legal departments manage the documents associated with their clients and cases. We ask our customers to trust us with their most sensitive documents and information, says Chris Finn, Cloud Architect at iManage. We need to prove that data's not only going to be secure and encrypted but also readily available whatever happens.

iManage software is available both on-premises and as a software as a service (SaaS) solution, run either in the company's own datacenters or on the Microsoft Azure cloud platform. This lets the company serve customers in geographies where it doesn't have the physical

infrastructure while helping ensure that data stays local and everyone stays compliant with privacy regulations. (Microsoft Corporation, 2019)

XYZ's stakeholders had made it clear that they desired not only an elastic and responsive platform however additionally and most importantly, a highly reliable infrastructure that Azure seemed to provide. iManage's story showed a clear picture of Azure's ability to provide a reliable infrastructure for a customer-facing SaaS application run by a small software development company. Additionally, not only was the platform trusted for its reliability however it also boasted a strong level of enterprise-grade security which the world had come to know from Microsoft.

To address the concerns of scalability of the application platform it was important to not only look at similar size SaaS migrations such as iManage but to look towards the company's future growth potential. Thankfully there is no shortage of confirmation in Azure's ability to scale on-demand as well as automatically with very limited human interaction required. An incredible example of this elasticity is shown through Azure's retail customers such as Forever 21, which utilizes the Azure platform to handle their e-commerce website day-to-day as well as through peak seasons such as Black Friday and Cyber Monday.

After the solution was deployed, the company's newfound agility instantly resulted in measurable benefits. When seasonal demand spikes struck, the company scaled up to as many as 120,000 concurrent sessions to meet demand. The result was unprecedented revenue. And after the holiday, Forever 21 scaled down from 64 web servers to just 16 servers, lowering costs and generating savings that it passes on to customers.

We had record Thanksgiving and Black Friday sales, says Diamond. With Azure, we were able to cope with more sessions and more online customers than we'd ever seen—by almost 50 percent. Ultimately, by scaling our presence in Azure, we got through our biggest holiday season ever without a hitch. (Microsoft Corporation, 2019)

As the investigation into the public cloud offerings continued management made it clear

that they desired a straightforward, manageable platform. If it can be done utilizing an automated process, do so. Thus, the implementation of utilizing the AWS platform would require a much more hands-on keyboard approach as there were not free utilities such as Microsoft's Azure Migrate automation tool, built specifically for migrating Windows and SQL workloads to Azure in as little as a few clicks of a mouse. Therefore, not to discount some of the key advantages of AWS, such as the more expansive product library, the final decision was to go ahead with Azure as the public cloud platform of choice because of the ability to rapidly rehost the existing SaaS application with simplicity and the added benefit of a lower monthly cost.

Project Rationale and Systems Analysis

As it is essential to have a proper design methodology, the Systems Development Lifecycle (SDLC) method will be utilized. The initial portion of which, planning, is undoubtedly paramount to the success of my project. Performing an in-depth system investigation of the current application's hosted server infrastructure allows the ability to identify any existing gaps. The outcome of the planning phase allows for the ability to properly enter the analysis phase such as performing an in-depth gap analysis. The Analysis phase is used to determine where the problem is, to fix the underlying system. This involves breaking down the system into different sections to analyze the situation based on project goals. Gauging what needs to be created to engage stakeholders for that definitive requirements can be defined. Once the requirements are set following the analysis stage the design can be created which will generally consist of system diagrams that may occasionally be created in a proof of concept environment. Once the design has been finalized and justification has been reached the implementation phase is entered. At which point in the lifecycle the final components are generated and deployed into the new environment for quality assurance testing followed by production deployment if performance is satisfactory.

To assess the impact of the problems, I spoke with those who were at the frontline and

feeling the pain most: the support technicians and the end-users. The support technicians spoke at an in-depth length to the abundance of support tickets being generated since the previously hosted server migration in-regards to poor performance and extended load times within the application. While the end-users I conversed with spoke with an equal discontent in-regards to not only the poor performance of the application in its current hosted environment however to their lack of faith in XYZ's ability to provide a reliable service referencing the outage back in April 2011. Customers I spoke with stated larger documents were now taking up to 10 times longer to download prior to the migration and that they were at times unsure if the service was even online or had once again gone offline due to underlying server connectivity or power issue.

To properly evaluate how the server configuration behaved, I commenced an in-depth review of current usage patterns, customer and application needs, and analyzed the architecture in its entirety for any possible root cause which could be impacting the performance of the application. This investigation was completed over two weeks alongside senior members of the development team during typical United States business hours of operation. This review allowed the team to see first-hand what our end-users were encountering. During the start of the day on the east coast and the end of the day on the west coast, speeds seemed to remain stable, though still far slower than had been present prior to the migration. Performing bandwidth performance tests which included sample file downloads through the application throughout the business day would allow us to see speeds as low as 100 Kbit/s at times while knowingly connected over a 300 Mbps fiberoptic network.

At the heart of the configuration, I began to look at a more in-depth level to the server utilizations themselves laying behind the bottlenecked network connection. Upon initial review, the findings were undoubtedly similar to those of the network bandwidth. During the beginning of the day on the east coast and end of the day on the west coast, the server utilization remained rather stable. However, throughout the day as more and more end-users connected the servers failed to scale effectively becoming unable to handle the increased workload of metadata

searches, updates, and additions through the SQL Server nor through application calls being made through the applications core Window's server service. As these two core functions began to fail, the team was able to watch as the systems slowed to a crawl causing an already limited network connection to be throttled further as the application server itself struggled to locate, request, and serve file download requests back to end-users in a timely manner. We could now safely deduce that these two core bottlenecks, the server's inability to scale and the networks limited bandwidth, were responsible for the core slowness the end-users were experiencing to their discontent.

Undoubtedly, XYZ needed to adjust course and migrate to a more reliable and elastic platform both in day to day operating speed for the application, as well as the business continuity which could be provided by utilizing Azure's more expansive and reliable cloud platform. Utilizing the Azure platform would allow for XYZ to resolve these key customer and technical support pain points while benefiting from an elastic and expansive platform trusted by companies and individuals around the globe. With product offerings such as Azure Virtual Machines to replace the physically hosted server hardware which allowed for single click expansion and increases of CPU, RAM, and HDD storage as well as Azure Virtual Networking which boasted bandwidth speeds of over 1,000 Mbps across a highly reliable network backbone, there seemed to be the ability to perform a seamless rehosting of the current SaaS back-end to the cloud. Additionally, the Azure platform includes multiple new offerings that could be utilized to modernize the document management system's server infrastructure including conversion from utilizing an Infrastructure as a Service (IaaS) during a rehost to a Platform as a Service (PaaS) offering following an application refactor.

Goals and Objectives

There are two main goals for this project. The first is to migrate the existing document management system server architecture from the private datacenters into Azure. The second is to

overhaul the existing server infrastructure to allow for further elasticity. The main objectives for this project include the following nine major points:

1. Develop a rehost design document
2. Configure proof of concept environment within Azure for testing and validation
3. Perform a complete data replication from the private datacenter to Azure
4. During a maintenance window cut-over production servers to the Azure environment
5. Develop a refractor design document
6. Configure proof of concept within an Azure QA environment
7. Perform a complete data replication between the production and QA environments
8. During a maintenance window cut-over production servers to the refactored environment
9. Meet deadlines and remain within the proposed budget of the project

In order to achieve the desired result, all objectives must be completed successfully. Each individual objective represents an integral part of the system migration and modernization for the hosted version of the document management system. Failure to complete any individual objective effectively would undoubtedly lead to the failure of the overall project.

Infrastructure documentation is crucial to the successful initial deployment as well as the day-to-day operations and maintenance of any application's back-end services. In today's world, it is often common in small companies for an application infrastructure to be maintained by a single individual who is the only individual who knows anything about the infrastructure's architecture. Should that individual decide to leave the organization or simply take personal time away from work, the organization may quickly find itself in a situation where they are left scrambling to find a resolution to a problem in-which they lack familiarity to even know where to begin to investigate. Thus, providing and maintaining accurate documentation and diagrams of each application's infrastructure is crucial for maintaining best practice and business continuity.

In order to confirm that the new application infrastructure provides the full functionality intended proof of concept environment would be created within Azure. Referencing the rehost

design documentation as a guide, an isolated environment would be built. Applying the same configurations as could be found in the live production environment hosted within the private datacenters, the team would be able to extensively and accurately test the cloud deployment. Through the utilization of a proof of concept for the new environment, XYZ will avoid any significant possible disruptions to the document management system's current end users. Employing readily available SQL scripts the development team will be able to stress-test the document management system's new application infrastructure generating mass amounts of application requests for metadata returns and updates as well as simulated customer document download requests to their local PCs connected over the internet to the application server. This quality assurance methodology can help to ensure the proof of concepts stability and performance as it would perform under comparable load in the existing production environment.

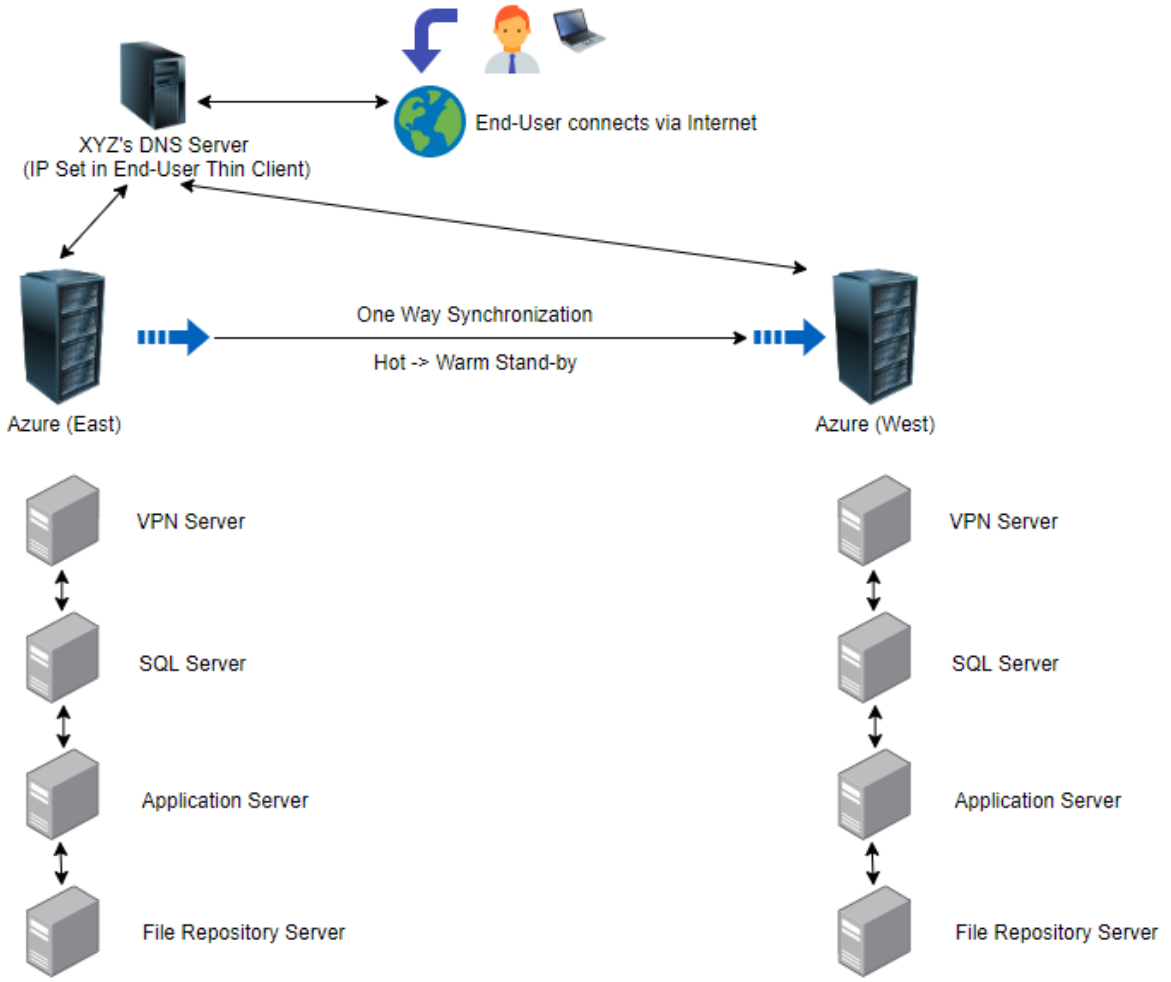


Figure 2. Diagram depiction of proof of concept application infrastructure in Azure (Rehost).

Once the proof of concept environment has been tested for stability, the customer data would be replicated from the active production infrastructure. As there are multiple portions, types, and massive quantities to the data it would need to be handled accordingly. Beginning with the Microsoft SQL Server data, each customer has their own SQL Database which would need to be backed up at the source and subsequently restored within the new SQL Server Instance running within Azure. Following next with the largest set of data, the customer file repositories which contain an encrypted version of every document and image stored within a customer's account. As there is a high risk of data corruption when copying files across an internet connection a block copy utility named RichCopy will be utilized. This utility utilizes a multi-threaded block-by-block copy approach which validates to ensure each individual byte of a file is copied successfully. Additionally, this utility allows for the scheduling of data replication and the ability to only copy items that have been changed, deleted, or added since the last data replication occurred. This functionality will be crucial in allowing for a final data synchronization during the maintenance window which will be utilized as a production cut-over period from the existing private datacenter infrastructure to the new Azure cloud infrastructure. Fortunately, the application servers themselves which control the application server services contain no individual customer data which needs to be replicated and can simply be created and validated ahead of data replication.

Once the rehost documentation is generated, the proof of concept has been tested for stability, and the data replication has commenced, the real challenge begins. To begin, a maintenance window would be scheduled at least two weeks in advance and customers of the hosted version of the document management system notified. Once notice has been given to the customers utilizing the platform the team would ensure that active data replication remains on-going at least once every 24 hours from the existing production environment to the new Azure infrastructure. This replication will ensure that once that maintenance window has commenced the data is in a recent state and a final replication which checks for only updated, removed, and added files can occur within a reasonable amount of time while customers are unable to access the

service. During this final file replication, the senior engineers assigned to the migration will stop the Microsoft SQL Server service rendering customers unable to access the application and utilize a basic T-SQL script to generate back-up files for each customer database. Once the back-up files have been generated, they will be copied via a secure connection to the new Microsoft SQL Server instance within Azure where a very similar basic T-SQL script will then be utilized to restore the customer databases. Following the finalization of the data synchronization, the SQL Server Service would be enabled within Azure and a small test period would occur by the dedicated senior engineers assigned to the migration.

Upon completion of the internal testing phase, the XYZ DNS server would then be updated to direct incoming application requests to the new Azure infrastructure. As the maintenance window concludes customers would be notified of the services renewed availability and the XYZ technical support team would enter a phase of hyper care in-which additional resource are available from the development and migration teams to ensure any issues which may be encounter by end-users are investigated and resolved in a timely manner. The hyper care period would last for two weeks followed by a return to standard day-to-day technical support operations. Following the return to normal business operations and a successful transition from the private data center environment to the new Azure application infrastructure, the project would move into the next phase in the process which focuses on the refactoring of the application infrastructure towards a more scalable, elastic, and futureproofed solution than the rehost alone can provide.

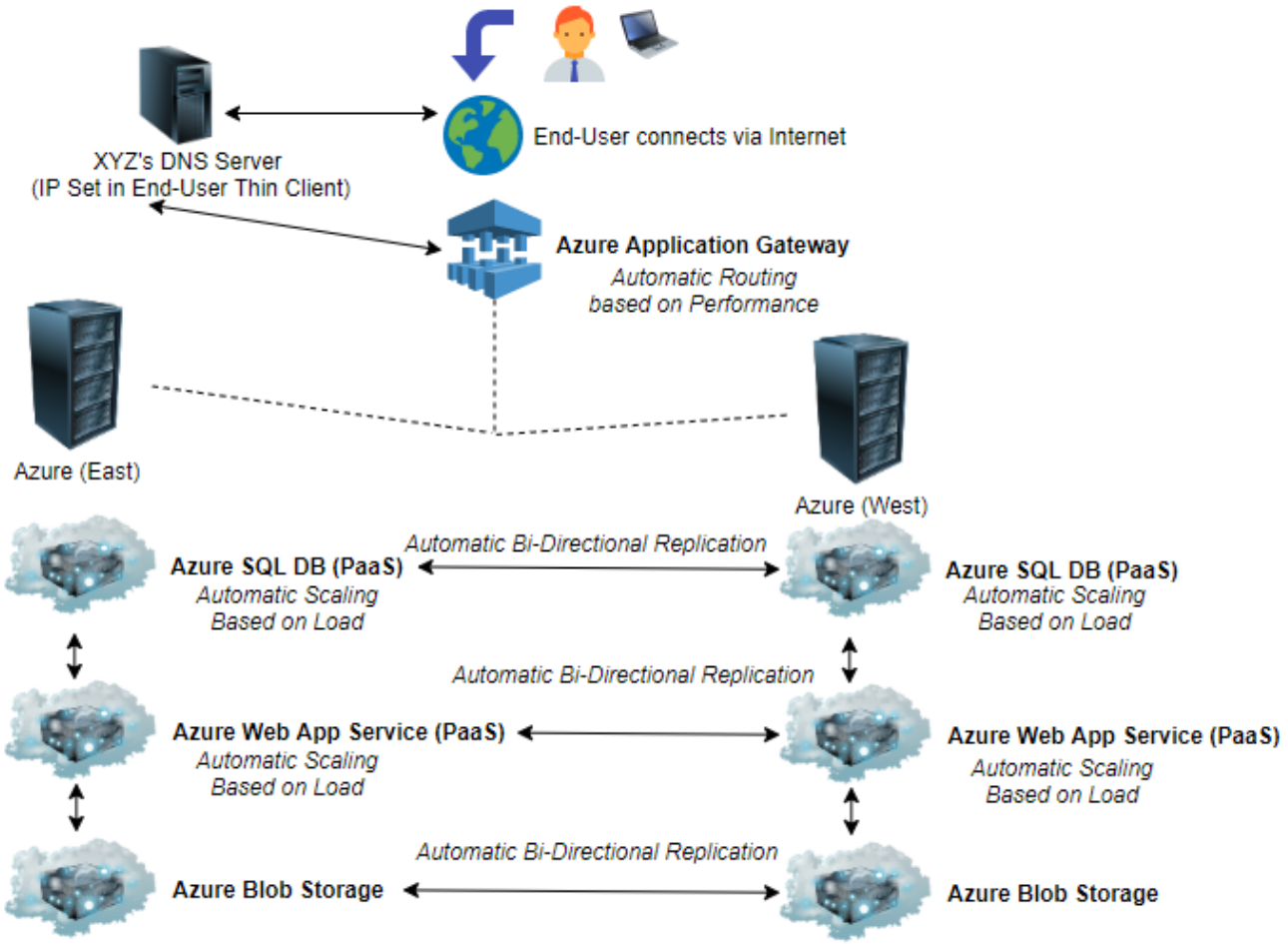
When reviewing the initial goals and desired outcomes of the project it is clear that rehosting the application infrastructure into a more reliable and more powerful IaaS solution is a great start, however, it does not allow for complete elasticity nor futureproofing of new bottlenecks which may occur. Microsoft’s own cloud migration journey guide states the first step is to rehost your on-premise or private cloud application to a public cloud platform, however, the next step if a company wants to truly utilize the power and benefit of the cloud is to refractor existing application infrastructures into a more elastic and scalable solution. (Microsoft Corporation, 2019)

This can be done by utilizing offerings such as Platform as a Service (PaaS) which removes some of the underlying infrastructure management associated with hosting an application. For example, when utilizing a PaaS solution, you no longer need to worry about patching the Operating System or other middleware that may be running on your server. Performing a review of the rehost design documentation alongside a list of Microsoft's Azure product descriptions it is clear there are a few wonderful refractor options which include the utilization of PaaS offerings.

Microsoft offers a hosted version of Microsoft SQL within Azure known as Azure SQL DB. By utilizing this PaaS offering it allows for traffic to be routed via an SSL connection over the internet, therefore, removing the need for a VPN client to be running on the end user's PC. This single refactor adjustment alone is a huge benefit and improvement to the overall customer experience as well as to the XYZ team. As the adjustment offers a reduced cost in monthly expenses to run the application infrastructure due to PaaS's lower cost to operate over IaaS as well as the removal of the need for VPN software licenses. Another PaaS offering within Azure known as Azure Web App allows for the ability to host .NET Services such as the Server Service behind XYZ's hosted document management system. Additionally, rather than run and maintain a repository server with the sole purpose of storing files Azure offers the ability to utilize storage accounts via Azure Blob Storage which can be accessed by simply redirecting each customer's repository location within their individual SQL Database which holds customer settings such as file repository paths.

By utilizing these PaaS offerings XYZ will be able to remove the need to maintain an underlying server infrastructure. Both in terms of physical hardware by now utilizing the public cloud Azure provides however additionally by no longer needing to maintain the Operating Systems, Middleware, or VPN. However, even with these incredible changes which allow for massively scalable elasticity both on-demand as well as through automatic triggers that can be easily configured within the Azure management console, the question of business continuity and performance remains. Thankfully, Azure PaaS services provide an incredible feature known as

data mirroring which automatically replicates data between two or more Azure datacenter locations of your choosing. Through enabling this automatic data replication feature of the PaaS services XYZ can then place an Azure Application Gateway as the new destination of redirection from the XYZ DNS server. This application gateway which acts as a load balancer can be configured in a multitude of ways however one of the most effective ways to utilize the benefit of the cloud and increase performance for the end-users would be to enable automatic routing based on performance. This method performs a quick ping test from the end user's PC to the possible PaaS service locations available, selecting the best performing with the quickest response rate and directing the customer's traffic to that location. (Microsoft Corporation, 2019) This newly refactored application infrastructure can be seen below.



With the above refactor design documentation in-hand the next phase in the project would be to re-enter a similar process as the initial rehosting of the platform to Azure

encompassed. To begin the project team will create a proof of concept for the refactored PaaS environment within Azure separate from the current production IaaS environment. Once the environment has been built, a test phase would commence where project team members utilize QA mechanisms such as T-SQL scripts to generate load testing like that of typical production workloads. Once the proof of concept has proven its ability to meet or exceed the currently in-place IaaS application infrastructure the process of customer data replication would begin. With data mirroring occurring a new maintenance window would be issued to customers two weeks in advance during which period data replication would continue to ensure a seamless and quickly executed final synchronization during the cut-over period. Upon reaching the scheduled maintenance window the team would disable the production servers from external access and execute a final data synchronization. While the final data synchronization occurs the project team of senior engineers will once again back up and restore the SQL Server Databases to the new Azure SQL DB PaaS instance. Upon completion of the file synchronization and SQL database restoration, the project team will run a brief test of all systems to ensure stability and desired connectivity. Following a successful test period, the project team will then ensure all services have been enabled and redirect the XYZ DNS server to now point to the Azure Application Gateway's IP address. As the maintenance window concludes customers would be notified of the services renewed availability and the XYZ technical support team would enter a phase of hyper care in-which additional resource are available from the development and migration teams to ensure any issues which may be encounter by end-users are investigated and resolved in a timely manner. The hyper care period would last for two weeks followed by a return to standard day-to-day technical support operations.

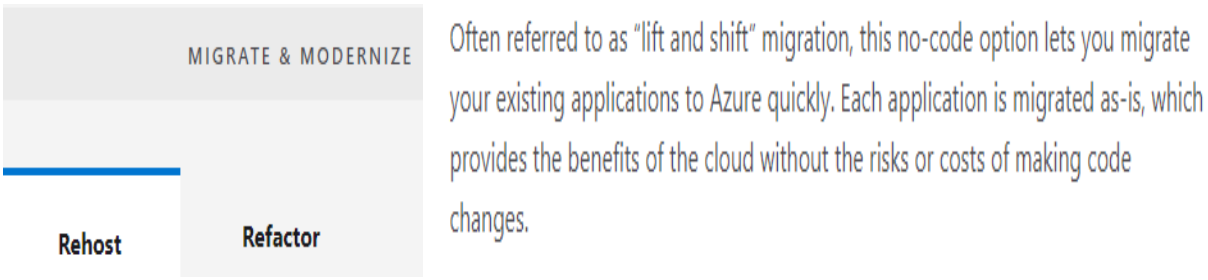
The final objective of this project would be ensuring that all activities are carried out within the budget and project timeline proposed. XYZ's leadership team has implemented a set budget for the entire scope of work in the project and it is within my objectives to meet or exceed these guidelines. Utilizing refactoring methods such as the conversion from a private cloud to public

cloud IaaS followed by a transition to a public cloud PaaS should substantially decrease the month to month operating costs of the application's infrastructure. As a tenured professional, producing solid results will guarantee that XYZ's leadership team is pleased with the result of this project.

Project Deliverables

The key deliverables for this project below encompass the result of all in-scope objectives and goals explained in further detail. As such, they include the migration and modernization of the document management systems application infrastructure to increase performance and reliability while decreasing the month to month cost of operation and increasing customer experience. Migration deliverables include design documentation which will include all documentation related to this project, including the business requirements, application requirements, private cloud architecture schematics, and specific design information including configurations, proof of concept QA results, and the IaaS rehost implementation plan. Modernization deliverables include design documentation which will include all documentation related to this project, including the business requirements, application requirements, IaaS architecture schematics, and specific design information including configurations, proof of concept QA results, and the PaaS refactor implementation plan.

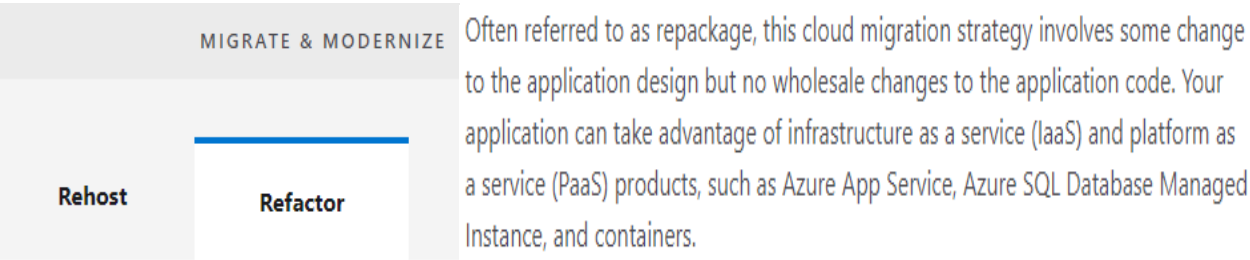
The application infrastructure migration is the foremost component of the project. We opted for the rehost then refactor method due to Microsoft's own Azure migration guidance methodology.



(Microsoft Corporation, 2019)

During the migration phase documentation will be completed which encompasses the configuration and architecture for the new application infrastructure within Azure's IaaS platform. This will allow the project team to ensure the day-to-day operation as well as expected and unexpected maintenance is able to be completed successfully with no individual XYZ employee as a single point of failure in the event of their departure from the company. Additionally, a well-architected overview of the IaaS platform architecture which encompasses how the infrastructure works together to support and run the application allows for quicker response time by the technical team in the event of an unexpected issue reported by an end-user.

The next component of the project focuses on the modernization of the application infrastructure. Again, we opted for the rehost then refactor method due to Microsoft's own Azure migration guidance methodology. Utilizing the rehosted application infrastructure within Azure's IaaS platform the application can then be refactored into a more modern application infrastructure utilizing the readily available Azure PaaS offerings.



(Microsoft Corporation, 2019)

During the modernization phase documentation will be completed which encompasses the configuration and architecture for the new application infrastructure within Azure's PaaS platform. This will allow the project team to ensure the day-to-day operation as well as expected and unexpected maintenance is able to be completed successfully with no individual XYZ employee as a single point of failure in the event of their departure from the company. Additionally, a well-architected overview of the PaaS platform architecture which encompasses how the infrastructure works together to support and run the application allows for quicker response time by the technical team in the event of an unexpected issue reported by an end-user.

Project Plan and Timelines

Timeline: 10/01/2018 – 03/19/2019

Project Deliverable or Milestone	Planned Duration	Actual Duration	Actual Start Date	Actual End Date
Project Phase: Planning				
Orientation	3 Days	3 Days	10/01/2018	10/03/2018
On-Site Meetings	2 Days	2 Days	10/04/2018	10/05/2018
Discussions with Stakeholders	3 Days	3 Days	10/06/2018	10/08/2018
Review Current Application Infrastructure	2 Days	2 Days	10/09/2018	10/10/2018
Receive Azure Demo	1 Day	1 Day	10/11/2018	10/11/2018
Receive AWS Demo	1 Day	1 Day	10/12/2018	10/12/2018
Investigate & Research Azure & AWS	5 Days	5 Days	10/13/2018	10/17/2018
Planning Phase Complete	0 Days	0 Days	10/17/2018	10/17/2018
Project Phase: Design - Migration				
Physical Design	3 Days	3 Days	10/18/2018	10/20/2018
Logical Design	3 Days	3 Days	10/21/2018	10/23/2018
Security Design	3 Days	3 Days	10/24/2018	10/26/2018
Design Phase Complete	0 Days	0 Days	10/26/2018	10/26/2018
Project Phase: Testing / QA - Migration				
Build PoC Azure IaaS infrastructure	10 Days	10 Days	10/27/2018	11/05/2018
Configure PoC Azure IaaS infrastructure	10 Days	10 Days	11/06/2018	11/15/2018
Initiate Data Replication	30 Days	30 Days	11/16/2018	12/16/2018
PoC Performance Testing	10 Days	10 Days	11/16/2018	11/24/2018
PoC Security Testing	5 Days	5 Days	11/25/2018	11/30/2018

PoC High-Availability / DR Testing	2 Days	2 Days	12/01/2018	12/02/2018
PoC Testing Phase Complete	0 Days	0 Days	12/02/2018	12/02/2018
Project Phase: Implementation - Migration				
Schedule maintenance window	14 Days	14 Days	12/02/2018	12/15/2018
Execute maintenance window	0 Days	0 Days	12/16/2018	12/16/2018
Execute final data synchronization	0 Days	0 Days	12/16/2018	12/16/2018
Perform system test analysis	0 Days	0 Days	12/16/2018	12/16/2018
Execute DNS Redirect to Azure IaaS	0 Days	0 Days	12/16/2018	12/16/2018
Enter Hyper Care Technical Support	14 Days	14 Days	12/16/2018	12/29/2018
Implementation – Migration Phase Complete	0 Days	0 Days	12/29/2018	12/29/2018
Project Phase: Design - Modernization				
Physical Design	3 Days	3 Days	12/30/2018	1/01/2019
Logical Design	3 Days	3 Days	01/02/2019	01/04/2019
Security Design	3 Days	3 Days	01/05/2019	01/07/2019
Design Phase Complete	0 Days	0 Days	01/07/2019	01/07/2019
Project Phase: Testing / QA - Modernization				
Build PoC Azure PaaS infrastructure	10 Days	10 Days	01/08/2019	01/17/2019
Configure PoC Azure PaaS infrastructure	10 Days	10 Days	01/18/2019	01/27/2019
Initiate Data Replication	30 Days	30 Days	01/28/2019	02/26/2019
PoC Performance Testing	10 Days	10 Days	01/28/2019	02/06/2019
PoC Security Testing	5 Days	5 Days	02/07/2019	02/11/2019
PoC High-Availability / DR Testing	2 Days	2 Days	02/12/2019	02/13/2019
PoC Testing Phase Complete	0 Days	0 Days	02/13/2019	02/13/2019
Project Phase: Implementation - Modernization				

Schedule maintenance window	14 Days	14 Days	02/14/2019	02/28/2019
Execute maintenance window	0 Days	0 Days	03/01/2019	03/01/2019
Execute final data synchronization	0 Days	0 Days	03/01/2019	03/01/2019
Perform system test analysis	0 Days	0 Days	03/01/2019	03/01/2019
Execute DNS Redirect to Azure PaaS	0 Days	0 Days	03/01/2019	03/01/2019
Enter Hyper Care Technical Support	14 Days	14 Days	03/02/2019	03/15/2019
Implementation – Modernization Phase Complete	0 Days	0 Days	03/15/2019	03/15/2019
Project Phase: Project Closing				
Design Documents Overview	1 Day	1 Day	03/16/2019	03/16/2019
New Architecture Review	1 Day	1 Day	03/17/2019	03/17/2019
Lessons Learned	1 Day	1 Day	03/18/2019	03/18/2019
Official Project Close-Out and Turnover	1 Day	1 Day	03/19/2019	03/19/2019
Project Completion	0 Days	0 Days	03/19/2019	03/19/2019

When comparing the planned duration against the actual duration of each item in the timeline it becomes apparent there is very little variation between them. The utilized timeline was heavily dependent on the senior engineers assigned to the project's ability to navigate and understand the Azure platform. As such, I allowed a reasonable amount of time during the initiation and configuration phases of the project during which engineers were able to come up to speed with the Azure product offerings. Once the team was knowledgeable and understood the Azure platform the execution of the migration itself during the maintenance window was able to conclude relatively quickly provided the groundwork was already laid effectively during the proof of concept phase. Overall, from initialization to close out, the actual time required for completion of this project was 169 days which matched the projects proposal.

Project Development

The SaaS migration and modernization project included several phases which lead to a successful completion. The project initiated with the planning phase which determined that Microsoft's Azure public cloud was the best fit for XYZ's present and future application infrastructure hosting needs. The planning phase additionally outlined the work which would be required, with specific goals, objectives, and deliverables outlined for an extremely accurate timeline estimate prior to beginning the migration and modernization. XYZ's cloud engineering team executed the action phase where the implementation of the work including design, testing, migration, and validation of the application infrastructure took place. During the preproduction phase the applications functionality was tested in its current form on the new platform. Any minor issues which occurred during the migration phase were then resolved during this period. This was followed by an additional action phase of modernizing the application within the new Azure environment and lastly by an additional validation phase to test the applications full functionality after modernization on the platform. The SaaS migration and modernization completed at this point with full success.

Problems Encountered

There were minimal problems encountered during the project, the few which were faced will be outlined below. After the initial migration of the SaaS applications repository file system, it was found that certain files were not copied over to the new system. The issue was identified by running a system file tree utility against the initial system and the new Azure environment to ensure bit by bit replication occurred. However, it was identified that certain files which were open by the SaaS application in the background were being held in a locked state. By simply purging the logs and adjusting the parameters in the file copy command we were able to copy the remaining files without recopying any non-adjusted since last sync files. This resulted in a quick resolution and complete satisfaction in the data replication process upon completion.

Unanticipated Requirements

There were no unanticipated requirements faced during the execution of the project. The project lead and engineers assigned all had extensive experience working with the SaaS application and its underlying dependencies. There would have likely been requirements which had gotten overlooked had these individuals not been involved in the project planning.

Reasons for Change

The project did not see any changes occur beyond the few minor issues which were encountered. None of the problems encountered caused a change in scope, timeline, or cost estimated to complete the project initially. Prior to writing the project proposal full stakeholder and expertise engagement occurred leading to a fully developed and extremely accurate project plan and execution.

Actual and Potential Effects

The realized effects of the project include a far more reliable and scalable infrastructure for XYZ's SaaS application. Additionally, the month-to-month cost of operation is greatly decreased while allowing for increased performance of the application and its underlying dependencies.

The potential effects of the project have wonderful upside and very little downside. The future growth of cost is almost entirely related to increased usage of the platform which would only be led by an increased end-user customer base which would in-turn provide an uptick in revenue to cover these costs profusely. XYZ will need to keep usage in mind however to ensure they are not generating resources without the realization of cost now that they are virtual infrastructures within the cloud. If XYZ was to launch their SaaS application to a much larger audience or expand indefinitely within its current environment there would be no roadblocks on the path ahead due to the modernization portion of the project onto an expansive and elastically scalable platform within the cloud.

Conclusion

The SaaS Migration and Modernization project was successful because it met and exceeded the needs of XYZ in providing a reliable, scalable, and cost-effective solution for their application infrastructure hosting. The final implementation diagram (see Appendix 1) outlines the scalable and modernized application infrastructure as expected for XYZ's implementation. This expertly designed architecture allows for almost zero required maintenance and minimal adjustments post-project.

The project was also successful because of the realized SLAs which where not improved and adhered to by Microsoft's Azure cloud offerings. If any of these SLAs were for any reason not met due to the underlying cloud infrastructure XYZ would now be reimbursed for these disruptions (see Appendix 2) which covered any unexpected credit costs they would need to provide to their end-users.

As with any business-critical application infrastructure plan, continuity of business is of upmost importance. XYZ outlined this requirement in the initial project proposal. Azure has allowed XYZ to utilize PaaS offerings which provide automatic replication between geo-redundant zones (see Appendix 3). This allows for complete business continuity of XYZ's SaaS application within the new environment as if one datacenter was to face even a complete shut-down from a natural disaster, within a moment of time with minimal interruption XYZ's application would be back online across the country.

XYZ's organizational goals are focused on providing as customer focused application to their end-users as possible. In the process, the company has managed to cut costs in their application infrastructure hosting as well as decrease the amount of labor spent by their engineers in managing the applications infrastructure. With Azure, XYZ has positioned themselves for an incredibly reduced potential of facing application downtime in addition to providing a completely elastic and scalable infrastructure for future growth and development.

References

Microsoft Corporation. (2019). Azure Hybrid Benefit. Retrieved December 27, 2019, from https://azure.microsoft.com/en-us/pricing/hybrid-benefit/.

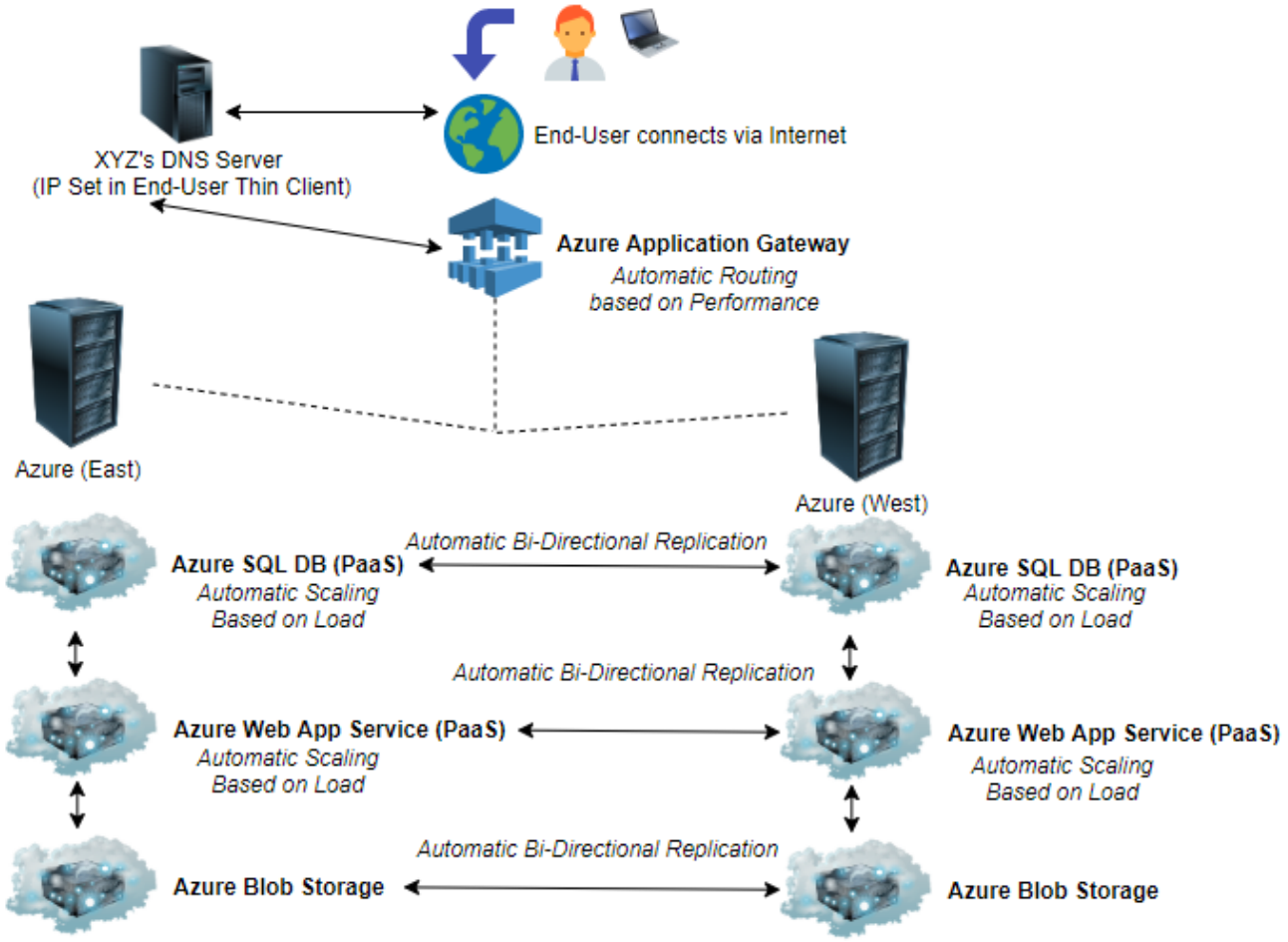
Microsoft Corporation. (2019). iManage boosts the competitive position of its SaaS offering with Azure. Retrieved December 27, 2019, from https://customers.microsoft.com/en-US/story/771911-imanage-partner-professional-services-azure.

Microsoft Corporation. (2019). Forever 21 moves to Microsoft Azure, register record-breaking holiday revenues. Retrieved December 27, 2019, from https://customers.microsoft.com/en-us/story/724149-forever-21-retailers-azure.

Microsoft Corporation. (2019). Cloud Migration Strategies: Microsoft Azure. Retrieved December 27, 2019, from https://azure.microsoft.com/en-us/migration/migration-journey/.

Microsoft Corporation. (2019). What is Azure Application Gateway. Retrieved December 27, 2019, from https://docs.microsoft.com/en-us/azure/application-gateway/overview.

Appendix 1: Final Implementation Diagram



Appendix 2: Azure SLA reimbursement policy

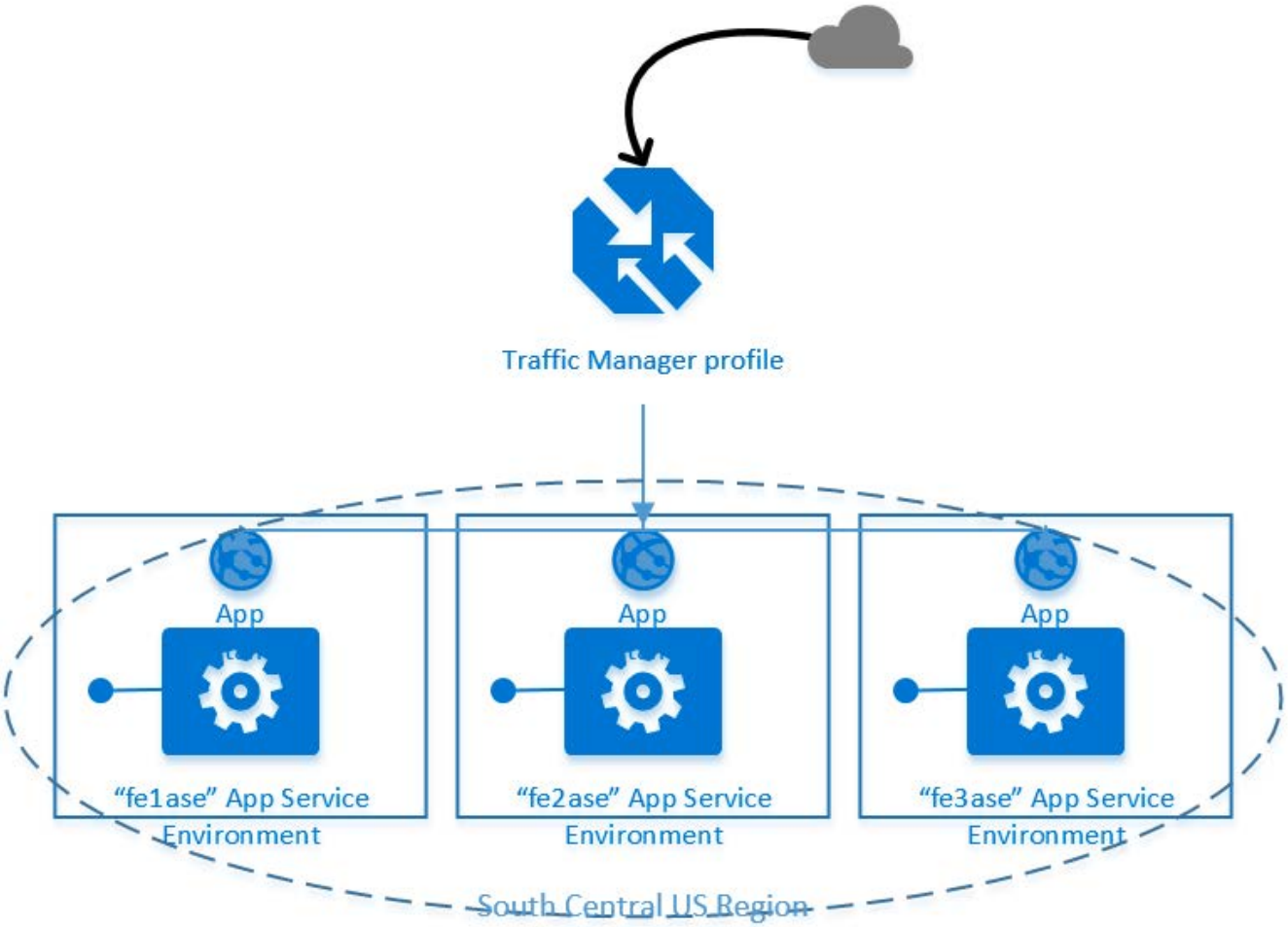
Service Credit

MONTHLY UPTIME PERCENTAGE	SERVICE CREDIT
< 99.95%	10%
< 99%	25%

The following Service Levels and Service Credits are applicable to Customer's use of the Business critical or Premium tiers of the SQL Database Service configured for Zone Redundant Deployments:

MONTHLY UPTIME PERCENTAGE	SERVICE CREDIT
< 99.995%	10%
< 99%	25%
< 95%	100%

Appendix 3: Azure App Service Geo Replication



Appendix 4: Competency Matrix

Domain/Subdomain	Competency	Explanation
Leadership and Professionalism	Communication and Interpersonal Skills	This report was written with strict adherence to the guidelines in which to cover the subject in question with accuracy and professionalism.
Language and Communication	Inquiry & Research	Throughout this proposal, I have explored scholarly articles as well as corporation, independent, and university research completed on subjects such as software development and the systems development lifecycle.
Language and Communication	Adaption	This proposal has been designed to adequately satisfy the needs of XYZ Incorporated.
Language and Communication	Written Communication Skills	Within this proposal, I have effectively shown that I can precisely and clearly organize my thoughts through detailed written communication.
Upper Division Collegiate Level Reasoning and Problem Solving	Problem Identification and Clarification	Throughout the development of my proposal, I recognized the many ways in which an application infrastructure can be organized and deployed across multiple platforms and virtualization methods while evaluating the best possible solutions available for XYZ's document management system.
Leadership and Professionalism	Organization Culture	The proposal has been crafted to tailor fit the culture of the company XYZ contains as a small software development firm.
Upper Division Collegiate Level Reasoning and Problem Solving	Reaching Well-Founded Conclusions	The project proposal includes empirical data on the process of platform migration and modernization using IaaS and PaaS offerings available on the public cloud.
Project Management	Project Planning	The crafted proposal incorporates the deployment methods utilized in the Agile methodology for the SDLC.

Language and Communication	Foundations of Communication	While writing this proposal I have applied the foundational elements of communication through the detailing of each subject in a clear, precise manner.
Quantitative Literacy	Utilize Standard Problem Solving Skills	During the process of implementation, many issues within the configuration require logical troubleshooting and in-depth research to resolve successfully.