Cloud Migration of Barclays' EPAS Platform

A Major Qualifying Project proposal submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelors of Science

By:

Kewen Gu, Computer Science (CS)
Patricio Janson, Management Engineering (MGE)

Submitted to:

Project Advisors:

Professor Michael Ciaraldi, Department of Computer Sciences

Professor Kevin Sweeney, Foisie School of Business

Sponsor Liaison:

Loren Taylor

Submitted on:

December 16, 2016

Abstract

The purpose of this project report is to serve as a reference for future developers, mainly from Barclays EPAS team, interested in migrating their applications from physical servers to cloud computing servers. The report was written after WPI students, Kewen Gu and Patricio Janson, investigated and tested several technologies and tools capable of facilitating the migration process and implemented a minimal application of cloud. In this project report you will be able to acquire information on cloud computing and its benefits, technologies and tools used for migration to the cloud, and instructions for testing a minimal application of cloud migration.

Acknowledgements

We would like to thank Worcester Polytechnic Institute and Barclays for letting us learn through this professional experience.

We thank the Barclays team for its hospitality and all the assistance they offered to finish this project; in particular, Neeraj Maruya, Manish Raikar, and Akshay Sardana. We would like to thank our sponsor liaison, Loren Taylor, for giving us the opportunity to put in practice what we have learned in college and assisting us in every matter since day one.

Finally, we want to thank our project advisors Professor Michael Ciaraldi and Professor Kevin Sweeney for taking the time to design this project in collaboration with Barclays and for their constant supervision that allowed this project to be completed.

Executive Summary

The EPAS team, in charge of maintaining a bespoke client reporting platform, expressed their interest in cloud servers as a cost effective alternative to physical data centers. Physical servers require a lot of infrastructure management time and a solution like cloud computing would allow developers to focus more on feature development and less on the maintenance of their servers. This interest originated since some of the EPAS physical servers are reaching the end of their lifespan.

Our job as consultants was to determine a feasible approach to migrate the EPAS platform from physical servers to the cloud by studying and testing different technologies and tools used in the cloud migration process. We also implemented a minimal application of cloud migration to make sure our approach was appropriate and secure. These practices allowed us to write and deliver a detailed cloud migration roadmap for the EPAS along with proper recommendations for the EPAS team's future cloud adoption.

Our technology research on the cloud and its business applications taught us that there is a common misconception that cloud is only used for storage. The cloud also allows us to run and manage applications on remote virtual servers provided by specialized cloud vendors. This technology is growing at a high pace and more companies from the financial industry are adopting cloud servers in their IT structure; however there might be some security concerns when sharing sensitive data with cloud providers. Cloud vendors like Amazon Web Services, offer professional and comprehensive support to eliminate these concerns and they are willing to collaborate with companies to develop safe security models to handle sensitive data. As part of our research we also compared the use of cloud and physical servers and concluded that cloud adoption can reduce companies' spending in IT and optimize their processes thanks to its ability to scale instantly and its effective hourly based cost strategies. The cloud also mitigates risks associated with hardware failure and natural disasters since applications and data are spread throughout different availability zones.

In order to put in practice the findings from our technology research and implement a minimal application of cloud migration the team went through a four step process:

1. Created a virtual machine to simulate a cloud environment. We initiated this process using Chef, a powerful but complex automation platform. To simplify our work, we decided to proceed with Vagrant to build and manage our virtual development environment.
2. Used Docker to pack and run applications as lightweight isolated containers on the virtual environment and automated the container creation process with scripts.
3. Validated the capability of the environment we created by hosting sample ASP.NET application as well as running Windows services inside the container. EPAS is written in .NET and it was necessary to use a tool called Mono to run .NET applications on the cloud.
4. And finally, we implemented a minimal application inside our environment using Space Manager, EPAS platform simplest service.

For the EPAS team to continue with this cloud migration process, we have carried out the following recommendations. First, we suggested investigating further on Chef and Mono's offerings, since we didn't have enough time to use and test all their functionalities. Due to existing concerns with some implementations of Mono, we suggested reconfiguring their existing authentication mechanism to adopt SSL or Kerberos. These network protocols provide strong authentication and encryption for client-server communications.

We also recommended using tools like Kubernetes for automating deployment, scaling, and management of containerized applications. Kubernetes can facilitate production efficiency and reduce management effort for systems with large amount of containers. As our last recommendation, we advised the EPAS team to start by implementing the simplest modules of EPAS on the cloud, and to experiment on the go, until gradually migrating the entire EPAS platform.

In conclusion, cloud technology matches the EPAS team's need as a cost effective alternative. Cloud technology will reduce their costs due to a better utilization of their resources both financially and computing wise and by explaining in detail the implications of migrating a the Space Manager service of this platform, the roadmap we delivered will work as a defined starting point to future development teams for the migration of the EPAS platform.

Table of Contents

Introduction

This section provides a brief description of the project's origin, goal, and the necessary steps that the team must undergo in order to successfully achieve the project's goal and satisfy Barclays' needs.

Origin

This project was done to fulfill the requirements of WPI's Major Qualifying Project. WPI's Wall Street Project Center revolves around the concept of "FinTech", which stands for Finance and Technology. WPI has designed interdisciplinary projects where students with business backgrounds and students with technical backgrounds can collaborate and work on technology solutions that aim to reduce costs, mitigate risks, and increase customer value for companies in the financial industry. In this case, the students were assigned to work with Barclays EPAS team, a software development team that manages a client-reporting platform.

As some of the Barclays EPAS physical servers reach the end of their lifespan, the EPAS team decided to look for a cost effective alternative to physical data centers. Cloud technology has raised interest to project sponsor Loren Taylor, head of the EPAS team. More companies in the financial industry are looking to implement cloud services to their technology infrastructure and it's logical to investigate the benefits of this technology in order to stay on competitive grounds amongst other financial firms. Several security concerns are raised given that companies in the financial industry rely on trust and any technology adoption must be meticulously studied before its implementation since companies of this sort handle sensitive data. The implementation of cloud services can reduce infrastructure management time and allow developers to focus more of their time on feature development, which can lead to an increase in customer value.

Goal

The team developed the following goal used to determine the project’s success:

Determine a feasible approach to migrate Barclays’ EPAS platform from physical servers to cloud servers.

Objectives

In order to achieve the project’s goal, the team set the following objectives:

- Study and test different technologies and tools that could aid the cloud migration process.
- Implement a minimal application of cloud migration using the technologies we´ve studied.
- Write a roadmap for Barclays’ EPAS to migrate their platform

These set of objectives were completed through an eight week term. By completing these objectives the team made sure that their approach to cloud migration was correct and that they could provide Barclays with a detailed set of instructions for future development teams interested in cloud migration.

Background

This section provides the reader with background information on the project, which will help the reader understand further sections of the project report. It also contains the team's technology research where you can get information on cloud computing and its services, the team's findings on the advantages of using cloud instead of physical servers, and technologies and tools commonly used for the cloud migration process.

Barclays

Barclays is an multinational corporate investment and personal finance bank founded in 1690 that is based in the United Kingdom. Barclays operates in Africa, America, Asia Pacific, Europe and Middle East in over 50 countries. The students performed the project in Barclay's New York offices.

EPAS

This is not the first Major Qualifying Projects that Barclays has sponsored for WPI. In 2015 a group of four WPI students enhanced the EPAS' Global Reporting Monitor (GRM) as their MQP project. EPAS is a platform that allows client services to generate customized reports based on the clients' needs. The GRM is EPAS' frontend user interface. The interdisciplinary team from 2015's MQP focused on improving the platforms' user interface in terms of self-sufficiency, which addressed workflow imperfections and resulted in several enhancements to the platform. The major enhancement was the development of a cloning feature which allowed client services to clone the characteristics of previous reports from different clients and provide a new report to a client with the same needs without needing IT to do this for them.

The EPAS platform was a project developed by the financial department of Barclays in order to generate bespoke financial reports for clients. As the platform grew as well as its functions, the IT team inherited the management of this platform. Since the handoff of this platform to IT, the EPAS team has focused on making the platform more user-friendly. They have achieved with the contribution of the 2015

MQP project, and now it is time for the EPAS team to focus on optimizing their technology infrastructure. As some of their physical servers are reaching the end of their lifespan they are looking for an alternative to physical servers and interest in cloud migration has been shown. It is the job of the 2016 MQP to advice the EPAS team on a feasible approach for cloud migration of the EPAS platform.

Technology Research

In order to advice Barclays on the migration process of the EPAS platform from physical servers to cloud servers the team conducted research on cloud computing and different technologies that could aid the migration process of the platform to the cloud. The following section introduces cloud services and describes several technologies that we encountered relevant to the server migration process.

Cloud Services

The majority of people might think that the cloud is only used to store data due to their use of some cloud storage service such as: Google Drive, Dropbox, or iCloud. This makes sense, and even though part of these users' definition is correct, the cloud is not only limited to storing data. We can start by defining what the cloud is. The National Institute of Standards and Technology's (NIST) defines cloud computing as: [Def.1]"a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". In more simple words cloud computing is a service that allows users to access, store, and manage applications and data through a network of remote servers without the need of physical computer servers. The concept of cloud computing has been offered to the public since the late 1990's with Salesforce.com's website delivery of enterprise applications. Cloud computing and its functions have grown significantly since Salesforce.com's first offering thanks to major contributors such as Amazon with its AWS products and the growing demand for this technology across different industries.

A common analogy used to compare conventional physical servers with cloud instances is the one of Pets and Cattle. Physical servers are like pets, who demand a lot of care. Pets also have a limited lifespan and the price to maintain them can be high. On the contrary cloud instances are compared with cattle. Cattle has a lower maintenance cost and its lifespan is irrelevant to the purpose they serve.

This rather young technology is growing at a high pace and reaching a point where companies in the financial industry are showing their interest on it however there is still a perception that the cloud might be insecure as financial firms might feel uncomfortable sharing control of their IT structure with cloud providers.

Cloud services compared to Physical Servers

Our findings showed several outstanding advantages of using cloud over physical servers. First, cloud is scalable. Developers can easily scale up and down its computing power based on clients' demand. As a result, IT Departments are able to maximize the utilization of computing resources requested on the cloud. On the contrary, physical servers require an upfront investment in order to prepare for the increase of future demand. And oftentimes, physical servers are running far below their full capacity. The key advantage of the cloud's ability to scale instantly allows IT departments to provide a stable service to its customers because developers can adapt to client's demand in no time and the process can be automated.

Probably the biggest advantage of cloud is its cost strategy. Cloud vendors charge their customers based on the customer´s use. This turns a capital investment, into a variable expense. The strategy is often compared to a taximeter, which makes cloud very cost effective. The lower cost of cloud computing is also attributed to providers' large economies of scale. As vendors customer base increase, it is easier for them to offer lower prices to their customers. On the other hand, physical servers are prone to cost more. Companies not only have to pay for the computing resources overhead, but also for the space, utility, operation, and maintenance costs. And if we go back to the problem of underutilization that comes when using physical servers, we can conclude that IT Departments pay for servers that are not being used. The cloud's hourly fares address this problem since developers have the possibility to shut down cloud instances when they are not being used and bring them back up when they are needed.

In addition, cloud can be more resilient and reliable than physical servers in terms of failure, mainly because cloud adoption mitigates significantly the risks associated with hardware failure and power outages. Cloud vendors also provide professional and comprehensive support, to assist their customers to prevent and mitigate risks. While on physical servers, IT departments have to face the risks by themselves, and sometimes the level of risk management is less robust and thus less effective. It is important to mention that IT departments must also invest in two different locations for their data centers to mitigate the risks of natural disasters and power failure. The two data centers must be on separate locations that are distant enough so that they can't be affected by the same natural disaster, also the data centers must run on different power grids. These data centers have diesel power plants to run for a few days, however, depending on the severity of a natural disaster, it can be impossible to resupply diesel if roads are closed. In 2012, Hurricane Sandy affected over eight data centers in the Manhattan area. The cloud solves this by distributing data and applications amongst several availability zones.

Another major setback that we found when using physical servers is that physical servers have a limited lifespan and the process to upgrade hardware tends to be very extensive. Said process can take months. The process starts with the approval of upper management to buy the equipment and ends when IT has the machines configured and running at the data centers. The upgrade process of physical servers along with the high levels of energy needed to run data centers can also be considered unsustainable. As more companies attempt to be more environmentally friendly, the adoption of cloud computing will reduce a company's amount of obsolete hardware.

The advantages previously mentioned can justify the EPAS team's interest of implementing cloud services to their technology infrastructure; however, the students didn't build a business case based on actual company numbers due to Barclays' policies and neither was this part of Barclays' needs.

Technologies & Tools

The following (section or graph) describes several technologies and tools that are associated to the cloud migration process and explains the functions of said technologies.

Docker	Docker is a platform to pack, ship, and run any application as a lightweight container. Docker containers wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries – anything that can be installed on a server. This guarantees that the software will always run the same, regardless of its environment.
Vagrant	Vagrant is a tool for building and managing virtual development environment. It provides easy to configure, reproducible, and portable work environments built on top of industry – standard technology and controlled by single consistent workflow to help maximize the productivity and flexibility of you and your team.
Chef	Chef is a powerful automation platform that transforms infrastructure into code. Whether it's operating in the cloud, on-premises, or in a hybrid environment, Chef automates how infrastructure is configured, deployed, and managed across your network, no matter its size.
.NET	.NET is a software framework developed by Microsoft. We can build many types of apps with .NET, such as cloud, IoT and games. Apps can run on Windows, Linux. Android, macOS, and iOS. And we can deploy them on servers or desktops or publish them to app stores.The EPAS platform is written in .NET framework.
Mono	Mono is a open source development platform based on the .NET framework, allows developers to build cross-platform applications with improved developer productivity. It includes both developer tools and the infrastructure needed to run .NET client and server applications.
Apache	The Apache HTTP server is the world's most used web server software. It is developed for modern operating systems including UNIX and Windows, and provides a secure, efficient, and extensible server that provides HTTP services in sync with the current HTTP standards.

Amazon Web Services

The EPAS team expressed their interest in Amazon Web Services (AWS); however, it's too early for them to use the services of AWS. Part of the student's job is to make sure that the EPAS platform is compatible with cloud services for future implementation with AWS, which is why the students simulated a cloud environment to test cloud migration of the EPAS platform. We must remember that Barclays handles sensitive customer data and that any technology must be studied and tested appropriately before its implementation. The implementation of minimal application of cloud migration is explained in detail in the Methodology section of the report. Brief research on AWS was done as it seemed appropriate to consider for further implementations. AWS website worked as a decent source for the students to understand cloud computing and its business applications.

As the leading competitor in cloud technology services, AWS offers a wide variety of products and tools for storage, development, management, and security. AWS offers plans that adapt to their customer needs and have established cost effective pricing strategies with hourly based fares. Economies of scale allow AWS to offer low cost for their cloud computing services since they have hundreds of thousands of customers around the world. AWS focuses on ease of transition for their clients, this helps address compatibility concerns since AWS supports a wide variety of programming languages and technologies. AWS is known to be user oriented since it hosts forums and client events to listen to the cloud-using community. We took a look at some financial companies who are using AWS as a cloud provider. Most of the financial companies are startups and young companies however some bigger companies are benefitting from the use of AWS.

AWS Client Testimonials

These are some client testimonials from AWS customers who belong to the financial industry. These testimonials show how other financial firms have used AWS to improve their businesses. The team considered that this might be worth sharing as these testimonials are other representations of the application of FinTech.

Capital One:

As one of the leaders on digital banking, Capital One has used AWS to reduce it data center footprint. Capital One claims to have used or experimented with nearly every AWS service for development, testing and building purposes. Their most recent use of AWS was to deploy their flagship mobile application. Rob Alexander, Capital One's CIO, claims that they have worked together with AWS to develop a security model capable of operating more securely in the cloud than in their own data centers. Capital One has benefited from AWS ability to deploy cloud instances on the fly, its ability to adapt to demand, and its high availability and reliability.

National Bank of Canada:

The National Bank of Canada's Global Equity Derivatives Group (GED) was looking for a more effective way to process financial data as it was having difficulties to scale their data analysis platform using physical servers. In order to analyze unstructured and structured data, the GED decided to develop a solution using Cloudera[1] and pilot test it in the cloud taking advantage of the cloud's deployment processes and AWS compatibility with their technology of choice. The adoption of AWS sped up GED's data analysis time.

[1] Apache Hadoop based software.[2] A hypervisor is a piece of computer software, firmware, and hardware which enables building and running virtual machines.

Methodology

Project Management

The goal of this project was determined based on the theory of SMART goals. A SMART goal is a goal that is Specific, Measurable, Achievable, Realistic, and Time-based. The team used these characteristics to determine a goal because we wanted to provide Barclays with a quality final product that was helpful for their future practices. We also took in consideration that in the process of developing a final product, the team could also learn from the MQP experience. We based our goal on the team member's qualifications and we ensured to administrate the eight weeks of the project's length to guarantee optimal utilization of the time we spent in Barclays. Refer to the Introduction section to read the project's goal and necessary steps to achieve it.

In order to complete this project the team implemented several practices that facilitated communication between management, WPI advisors, and the students. Constant communication with team manager, Loren Taylor, was crucial for the completion of this project. Loren Taylor was very responsive to the team's email and phone calls, which involved him more in the development of the project. Weekly conference calls with WPI project advisors occurred on Tuesdays. In these meetings, agendas were provided in advance by the team so that attendees could prepare in advance. The meetings were held in Barclays' conference rooms which allowed Loren Taylor to participate of these meetings. In the meetings, the team would state their project status, completed items, future tasks, and specific points of discussion relevant to the point in time when the meeting was held.

The team created a project timeline and a critical path diagram to manage the project's progress and complete tasks on time when they were assigned. The timeline was updated when tasks were added or completed. The timeline was divided in 8 weeks based on the length of the team's stay at Barclays. Refer to Appendix A to see the final version of the project timeline that the team used. The critical path diagram was used as a complement document to the project timeline to define the importance of each task and how some tasks relied on others to be completed. Refer to Appendix A for an illustration of the critical path diagram we created.

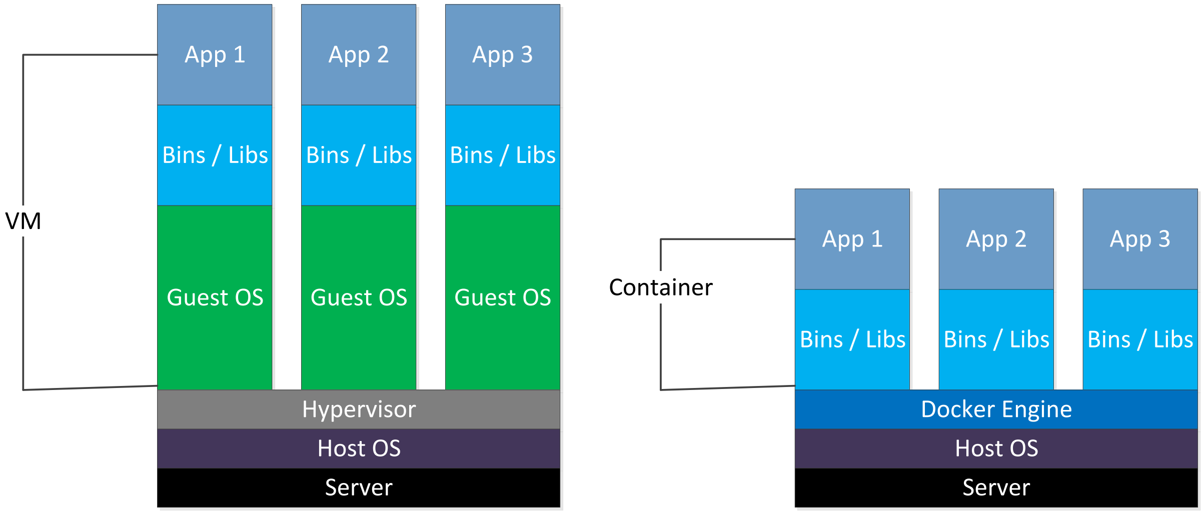
Process

This section discusses the process and steps the team has been through to try out different tools, to create Docker containers, and to run part of the EPAS platform in virtual environment created. Step by step instructions can be found in the roadmap on Appendix B.

Technology Investigation Process

At the beginning of this project, the team was advised to work on one of the Barclays' internal cloud offering, which is architected as an internal physical server running as a hypervisor[2]. The team began with studying and investigating different tools and technologies. During this investigation process, the team familiarized with two technologies: one is virtualization, and the other is containerization. The distinct architectures of these two technologies are shown in Figure 1.

Figure 1 – Virtualization vs. Containerization



With virtualization, each virtual machine runs its own entire operating system inside a simulated hardware environment provided by the hypervisor running on the physical hardware, while

[2] A hypervisor is a piece of computer software, firmware, and hardware which enables building and running virtual machines.

containerization allows multiple applications to run in isolated partitions of a single Linux kernel[3] running directly on the physical hardware. The entire user space environment is exclusively presented to the container, and any changes to it do not impact other containers' environments. To provide this isolation, a combination of OS-based mechanisms is used: Linux namespaces[4] are used for isolation and scoping mechanism. File system[5] mounts define what files are accessible to the container, while cgroups[6] define resource consumption of containers. Still all containers share the same OS kernel which can realize memory footprint efficiencies when identical libraries are used by multiple containers.

[3] The kernel is a computer program that constitutes the central core of a computer's operating system. It has complete control over everything occurs in the system.

[4] Namespaces are a feature of the Linux kernel that isolates and virtualizes system resources of a collection of processes.

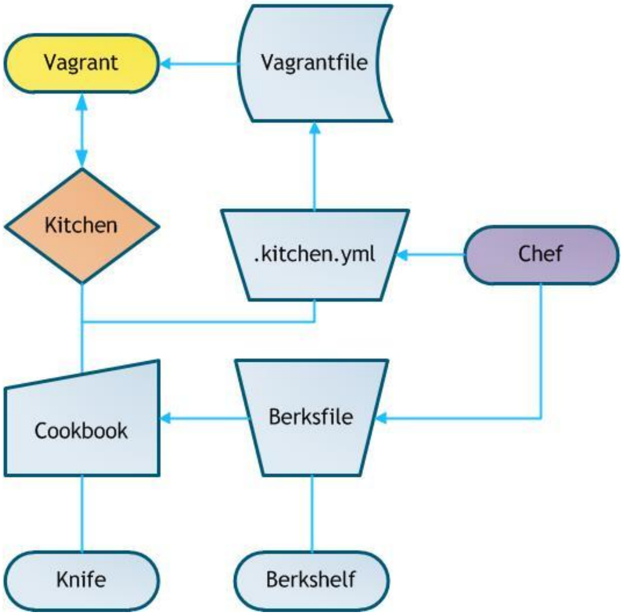
[5] In an operating system, the file system is used to control how data is store and retrieved.

[6] Control group (cgroup) is a Linux kernel feature that limits, accounts for, and isolates the resource usage of a collection of processes.

Virtual Machine Creation Process

In order to simulate the cloud environment on the hypervisor, the team had to create a virtual machine, and then create containers inside the VM to host the applications. The team was advised to follow the instructions on Barclays' internal confluence page for creating and setting up Test Kitchen[7]. The team found that Chef is a powerful tool for automating the configuration process of creating virtual machines, while Vagrant is a tool used predominantly for creating virtual machines across different platforms. After successfully creating the Test Kitchen, the team was able to host a simple http server inside the kitchen using python. The workflow of Chef inside the hypervisor is shown in Figure 2.

Figure 2 – Workflow of Chef inside the hypervisor



Although Chef is powerful and utilizes additional tools such berkshelf[8] and knife[9], the team found it to be less straightforward than using Vagrant, since it involves the integration of different tools together which the team would barely make use of. As a result, the team was advised to solely use Vagrant to

[7] Test Kitchen is a virtual environment created and provisioned by Chef with the help of Vagrant.
[8] Berkshelf is a dependency manager for certain cookbook workflows that is included in the Chef Development kit.
[9] Knife is a command-line tool that provides an interface between a local chef-repo and the Chef Server. It helps users to manage different kinds of data.

create virtual machines. The process for creating VM with Vagrant was completed with ease, since it shares similarities with the creation of Test Kitchen using Chef. During the creation, Vagrant pulled a VM base image from Barclays' internal storage. Next, Vagrant set up the shared ports and folders between the guest and the host. Thereafter, the team was able to ssh[10] into the newly created VM using Vagrant's default ssh key.

Docker Containerization Process

In order to install Docker inside the VM, the team had to use yum[11] to pull the Docker packages from Barclays' internal yum repositories and then install the packages. Before running Docker, the team had to run the Docker daemon[12] which was installed along with Docker. After Docker was configured and fully functioning in the VM, the team wrote a script to automate the container creation process. The team used mono in order to run EPAS on Linux because EPAS is written in .NET framework, which only runs on native Windows OS. In order to install mono using the script, first it was necessary to replace the default yum repos of the base image with Barclays' internal yum repos. Second, we had to modify the network and login settings. Next, we added mono, xsp[13], mod_mono[14], and libgdiplus[15] source packages into the container. Then, we used yum to install all dependencies which are required for mono to compile and run. At last, we needed to compile and install mono, xsp, mod_mono, and libgdiplus. Therefore, the team was able to install mono, and all the dependencies in order to run .NET applications on Linux based containers. Once containers can be created and replicated using the scripts, the team completed the first part of the project, which was to investigate and test tools that were able to create a containerized

[10] Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. The best known example is for remote login to computer systems by users. This is also the method how the team login to the hypervisor.

[11] yum is a software package manager that installs, updates, and removes packages on RPM-based Linux systems.

[12] The Docker daemon is the persistent process that manages containers.

[13] XSP is a simple, standalone web server written in C# that can host ASP.NET's system for Linux and other Unix operating systems.

[14] mod_mono is an Apache 2 module that provides ASP.NET support for the web's favorite server, Apache.

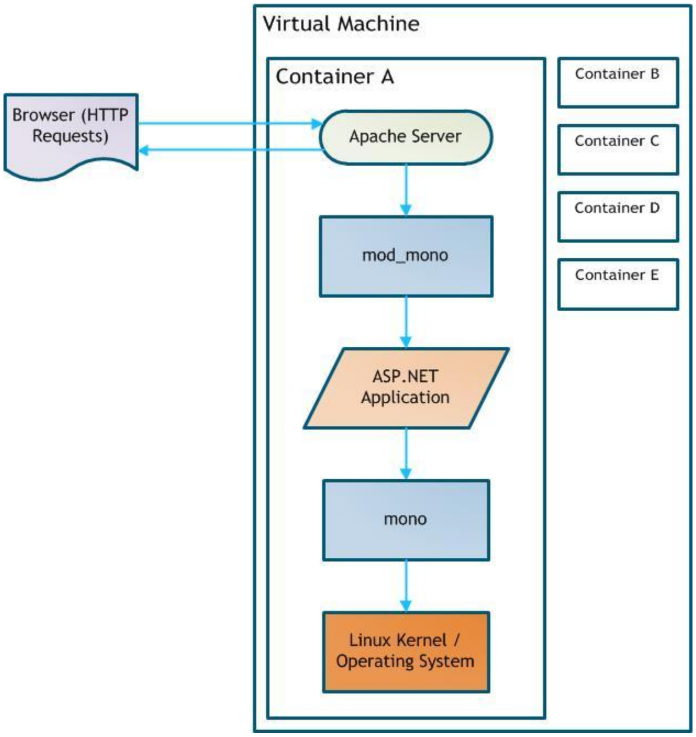
[15] libgdiplus is the mono library the provides a GDI+ compatible API on non-Windows operating system.

environment suitable for the EPAS platform. The next step was to test sample .NET applications in the containers.

Container Verification Process

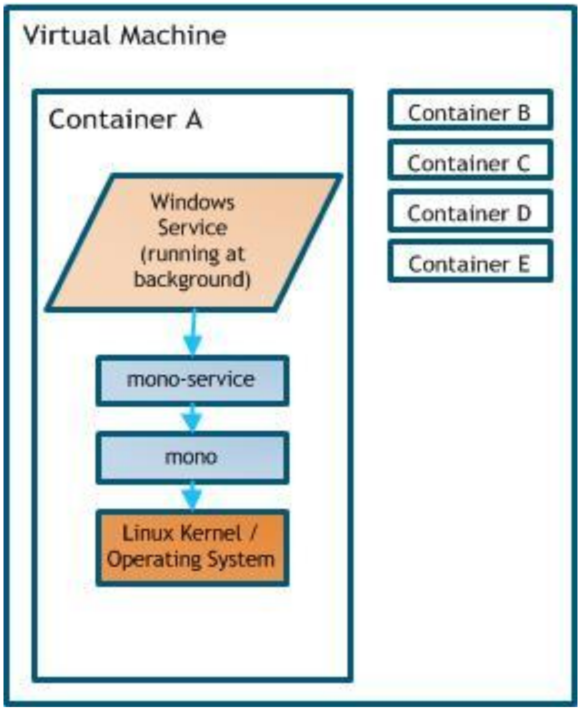
The team hosted an ASP.NET[16] sample application, and tested to run Windows services in the container to verify the capabilities and limitations of the Linux virtual environment. Results showed that our environment is competent to run both ASP.NET application and Windows services. The visualization of how this process works can be seen in Figure 3 and Figure 4 below.

Figure 3 – Workflow of running ASP.NET application inside the container



[16] ASP.NET is an open-source server-side web application framework developed by Microsoft as a part of .NET framework, and designed for web development to produce dynamic web pages.

Figure 4 – Workflow of running Windows Service inside the container



Inside, mono works as an intermediate layer that enables .NET applications to run on top of the Linux kernel. For running ASP.NET applications, mod_mono is required to connect the application with the Apache 2 server. Therefore, when the client's browser requests for the ASP.NET application, mod_mono makes sure that the server knows what to respond to the client. Afterward, the team found a tool called mono-service which is provided by mono for running Windows Services in Linux. Window services can perform a variety of jobs while running on the background, such as file monitoring, database handling, report generating, etc. They are also widely used by our EPAS platform.

At this stage, the team had completed the process of containerizing a .NET development environment in the Linux virtual machine, and was able to repeat this process using our customized scripts. After containers were validated for running .NET applications, we migrated SpaceManager, which is a service of EPAS, and implemented it in our cloud environment.

Minimal Application Implementation Process

The SpaceManager module of our EPAS platform is the minimal application we chose to test inside the container we created. It is basically a persistent application running as a Windows Service that monitors all our clients' files stored in the database. When an individual file reaches its TTL (Time to Live) specified at its creation, SpaceManager would remove it, and clean up the database storage to make room for other incoming files. There're two programmatic jobs that we were not sure of on how mono implements them on Linux. First is how mono writes logs, and the second is how mono connect our application to the Microsoft SQL Server if such connection is required.

To verify our concerns, we wrote two Windows service applications. One was to solely output a timestamp every 30 seconds and write this to a local log file. It turned out that we can specify the path of the log file in the code, and mono would write logs directly to that path. However, if a valid path is not provided, mono requires an environment variable to be set, so that it knows where to output the log. The second window service we wrote was solely to connect to the Microsoft SQL Server. Our verification results showed that mono cannot use the Windows build-in authentication method, since it's running in Linux. Mono does provide a method for establishing the connection to SQL Server, which requires the users to manually input their user ID and password explicitly in the code. However, this method can never be acceptable at production level, since it is insecure and prone to risks. In order for our cloud implementation to meet our security standards, we tried to investigate other solutions around this database connection and authentication issue.

Challenges

This section mentions the difficulties that the team faced when working on this project**.** The team faced minor challenges through the project and most of them were related to Barclay's security policies which are imposed to protect their customers' privacy. This is understandable but it is important to report them because these challenges create non-value added time. These are time consuming and it is something that the EPAS team should have in consideration for future practices especially when they are working with strict deadlines.

Limited privilege:

Inside the hypervisor, the team's access was restricted. This is to protect the hypervisor itself from any misusage caused by individual users of the system. As a result of this, the team was not able to install any tools and frameworks on the system. This wasn't a problem anymore after we successfully created a virtual machine inside the hypervisor, since we are offered full access to the VM.

Restricted network access / Firewall:

Inside the hypervisor, though it was possible to connect to Barclays' intranet, any access to outside network was forbidden. However, we were offered a proxy address by the IT engineering team to download any necessary packages from the worldwide web.

Limited internal offering:

Barclays' internal yum repositories only offer limited packages and many of these packages are outdated; therefore, the team had to compile and install some packages from source, which added extra work and time.

System and software version compatibility

The outdated packages caused capability issues for us along the way of our try out, both between packages and between the package and the system.

Technical Limitations

This section discusses some limitation of using Linux based cloud offerings to run and host .NET applications.

.NET is a framework developed by Microsoft, and originally solely supports Windows operating system. It's integrated into Windows environment, which makes it overwhelmingly convenient and user-friendly inside Windows. However, due to this factor, .NET possesses several drawbacks when running in non-Windows environment. The drawbacks are described as following:

- .NET is isolated. It runs primarily on Microsoft Windows. In order to run it in Linux, Mac OS, or other operating system, tools like mono need to be used to add a link layer between .NET applications and other operating system's runtime/kernel. Even though running .NET on other platforms is possible, another issue outstands with the lack of resources and lack of supports. It's easy to find solutions for problems with .NET encountered on Windows, since there's a very large community online which actively uses and maintains it. It might be difficult to find support or instructions, online or offline, because not many people have the need to run .NET applications on other platforms.
- .NET is not free. Although Microsoft has made the .NET core open-source recently, it's still costly compared to other open source software and programming languages. Furthermore, because .NET only supports Windows, the company has to purchase other Microsoft products in order to run and use .NET, such as Windows Server, Visual Studio, and so on. The price that comes with .NET could be staggering, but it is worth the price if the application is running on Windows.

The team has discussed the advantages of cloud, which was part of our project. One important factor of cloud is that most of technologies it uses are open-source, which means a framework like .NET cannot get much support from it. That's the reason we're using mono to run .NET applications on the cloud. However, the team concerns about the capabilities and limitations of mono as a middleware to run .NET in Linux environment. The limitations of mono and the cloud include but not restrict to:

- Windows Internet Information Server (ISS) is used to host ASP.NET applications. While on cloud, because of its Linux offering, Apache HTTP server is used instead. Both IIS and Apache are used widely and share a large portion of the market. Our concerns rise because IIS, just like .NET, is a platform designed primarily for Windows. And we're not sure whether Apache is capable of doing everything we can do on IIS, which requires further investigations.
- On Windows environment, authentication is simple, because everything can be authenticated by user's Windows account and their group permissions. However, Linux isn't designed to provide such convenient mechanisms. Therefore, it is necessary to investigate methods for the original mechanism to work on the Linux environment, or to modify the current mechanism to use other authentication methods or protocols, such as SSL or Kerberos.

Our concerns more or less are related to potential compatibility, security issues with switching between platforms and tools. However, we believe that comparative tools, such as IIS and Apache, share the same usages and characteristics. In order to be used so vastly, which these tools already are, they must have the ability to fulfill all of Barclays' needs.

Achievements

The team was able to deliver a roadmap to the EPAS team on time thanks to proper technology investigation and their success at implementing a minimal application of cloud migration. The minimal application consisted of uploading Space Manager, one of the EPAS platform's most simple applications, to a cloud environment. A walkthrough meeting was held so that the team could go through every step of the roadmap with the EPAS team to assure a complete understanding of the necessary steps to implement a minimal application of cloud migration. The roadmap was uploaded to the EPAS team confluence page and will work as a reference for their future practices. Refer to Appendix C to read the roadmap delivered to the EPAS team.

Recommendations

The team deliberated the advantages and limitations of the cloud, and carried out recommendations for Barclays' future implementations of cloud migration. From what we have investigated so far, we are optimistic about cloud migration for our EPAS platform, although we still have security concerns on some implementations of mono. Based on our research, mono is an excellent tool which enables running .NET applications on Linux. However, its implementations of some programmatic tasks may completely differ from what we normally do on Windows. And because of the limitation of running .NET on Linux, some of the native .NET/Windows features still cannot be provided by mono. As of now, mono can do most of our requirements, but sometimes in an insecure way, such as the method used when connecting to the Microsoft SQL database server from inside the containers, for which we have to specify the User ID and Password explicitly in the code.

The team considers this project as an initiative of Barclays' cloud adoption plan. Due to our limited time here at Barclays, we're not able to fully implement the EPAS platform on the cloud. We would like to carry out some recommendations for the EPAS team to continue with this cloud migration process. First, we suggest investigating further into Chef and mono's offerings, since we haven't had chance to use and test all their functionalities.

We would also suggest reconfiguring our current authentication method to adopt SSL or Kerberos. These are network protocols provide strong authentication and encryption for client-server communications, and works well on both Windows and Linux.

In addition, we suggest using tools such as Kubernetes and Docker Swarm for automating deployment, scaling, and management of containerized applications. These tools aim to facilitate production efficiency and reduce management effort in a system with large amount of containers.

Finally, we suggest starting by implementing the simplest modules of EPAS on the cloud, experimenting on the go, and gradually migrating the entire EPAS platform.

Conclusions

After studying and testing cloud migration technologies and implementing a minimal application of cloud migration with Space Manager we have reached to the following conclusions:

- Cloud technology has developed enough for its implementation in financial corporations and security concerns can be resolved through proper cloud architecture and collaboration with cloud providers in order to develop effective security models.
- It is most likely that the adoption of cloud will reduce Barclays' technology infrastructure spending due to its effective hourly based cost strategy. It will also reduce infrastructure management time and will lead to an optimal utilization of the company's resources.
- It is possible to run and host .NET applications in a Linux based cloud environment with the use of proper tools however there are still some limitations like we expressed in our Technical Limitations section and the installation of these tools adds burden to the process of cloud migration.

- Projects like this allow students to put in practice the lessons they have learned throughout their career. Through this experience we learned work ethic since we were treated like consultants rather than students. Our problem solving skills were enhanced as we encountered new challenges during the project and we were able to learn from each other's area of expertise.

Works cited

Calamari, Ning, Tang, Thomas (2015): The Global Reports Monitor Enhancements

Doran, G. T. (1981). There's a S.M.A.R.T. way to write management's goals and objectives. Management Review, 70, 35.

Lawlor, K. B. (2012). SMART Goals: How the application of SMART goals can contribute to achievement of student learning outcomes. *Developments in Business Simulation and Experiential Learning*, *39*.

Menzel, M., & Ranjan, R. CloudGenius (2012): Decision Support for Web Server Cloud Migration.

[Def.1] Mell, P., & Grance, T. (2011). The NIST definition of cloud computing.

References

https://aws.amazon.com/?nc2=h_lg

https://www.docker.com/what-docker

http://kubernetes.io/docs/whatisk8s/

https://web.mit.edu/kerberos/#what_is

http://info.ssl.com/article.aspx?id=10241

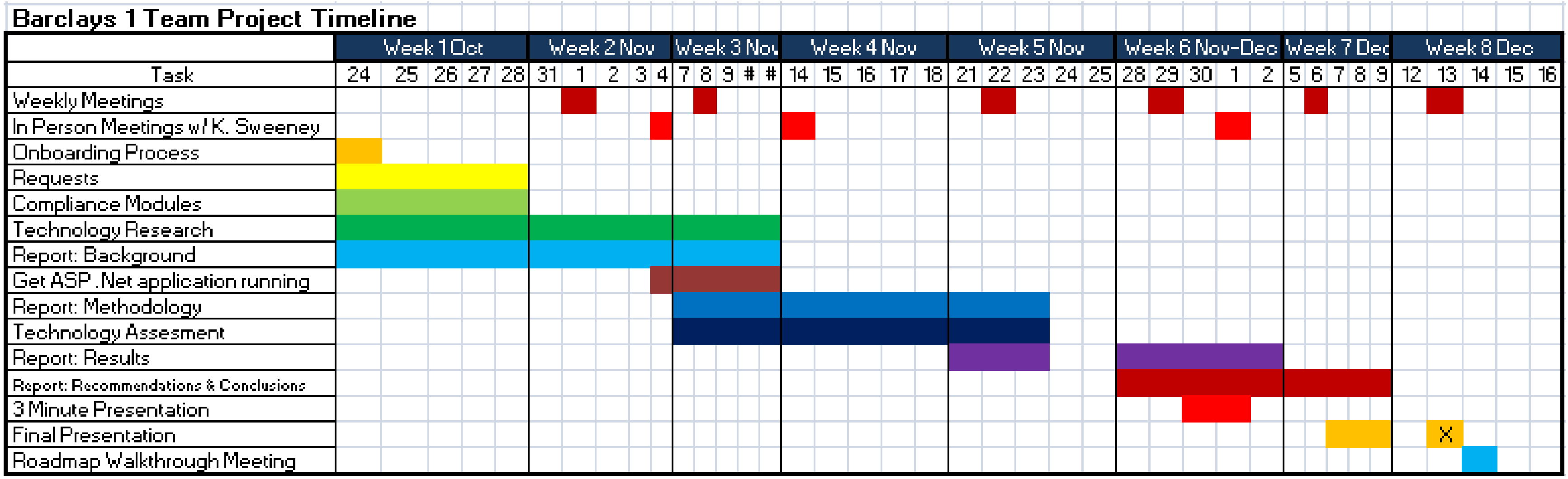
https://web.wpi.edu/Images/CMS/Biomedical/MQP_Project_Guide_December_6_2010.pdf

https://aws.amazon.com/financial-services/customer-stories/

Appendices

Appendix A: Project Timeline & Critical Path Diagram

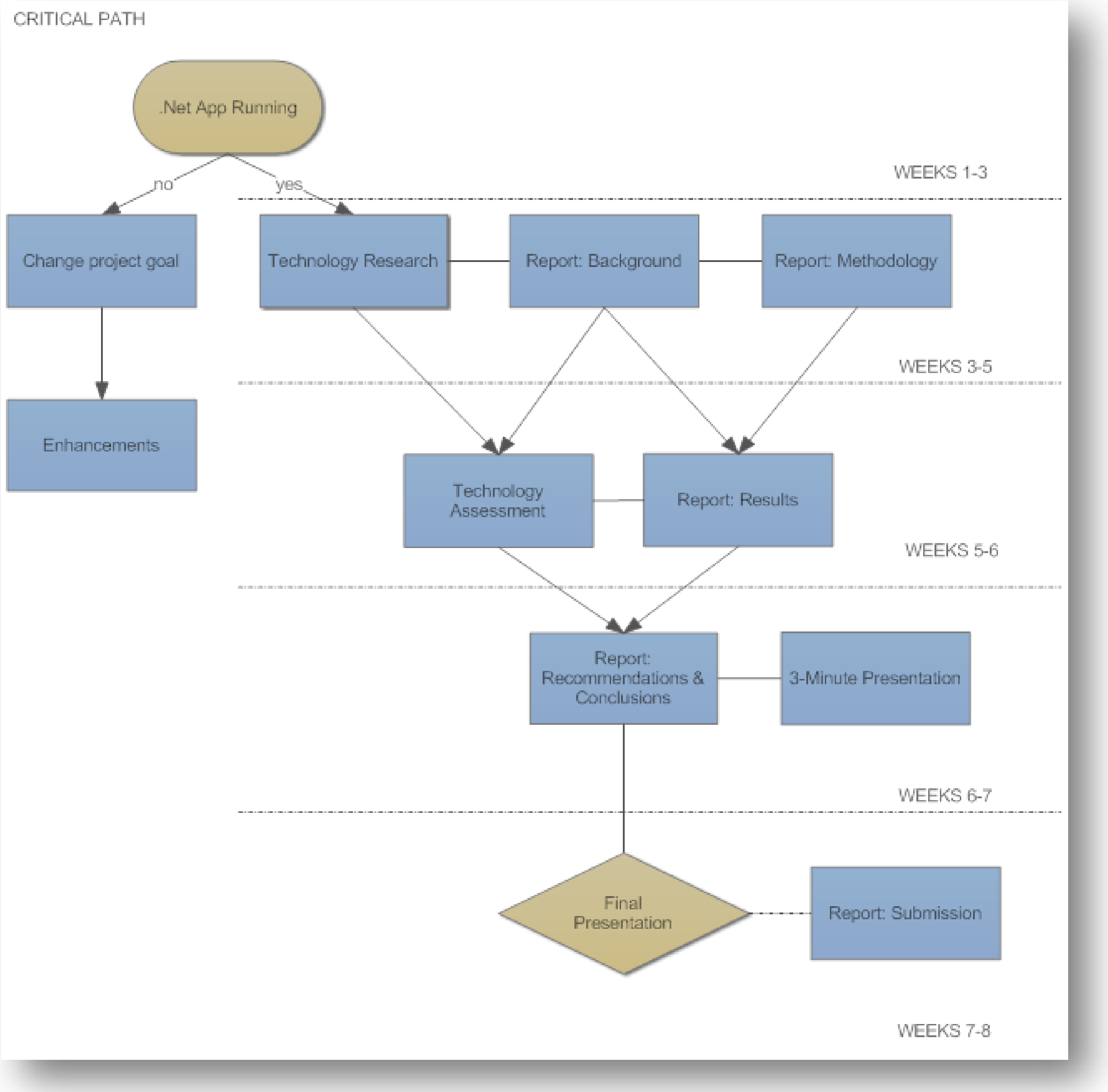
Project Timeline



Barclays 1 Team Project Timeline

	Week 1 Oct					Week 2 Nov					Week 3 Nov					Week 4 Nov					Week 5 Nov					Week 6 Nov-Dec					Week 7 Dec					Week 8 Dec				
Task	24	25	26	27	28	31	1	2	3	4	7	8	9	#	#	14	15	16	17	18	21	22	23	24	25	28	29	30	1	2	5	6	7	8	9	12	13	14	15	16
Weekly Meetings							🟥					🟥										🟥					🟥					🟥					🟥			
In Person Meetings w/ K. Sweeney										🟥						🟥													🟥											
Onboarding Process	🟧																																							
Requests	🟨	🟨	🟨	🟨	🟨																																			
Compliance Modules	🟩	🟩	🟩	🟩	🟩																																			
Technology Research	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																									
Report: Background	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																									
Get ASP .Net application running										🟫	🟫	🟫	🟫	🟫	🟫																									
Report: Methodology											🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																	
Technology Assesment											🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																	
Report: Results																					🟪	🟪	🟪			🟪	🟪	🟪	🟪	🟪										
Report: Recommendations & Conclusions																										🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥					
3 Minute Presentation																												🟥	🟥											
Final Presentation																																	🟧	🟧	🟧		X			
Roadmap Walkthrough Meeting																																						🟦		

Critical Path Diagram



Appendix B: Roadmap for implementation of minimal application

Cloud Migration Road Map

This page serves as a road map for EPAS's cloud migration strategy.

Introduction

Starting from October 2016, a group of two undergraduate students from WPI (Worcester Polytechnic Institute) came to Barclays to complete their MQP (Major Qualifying Project). They were advised by Loren Taylor to work on EPAS's cloud migration initiative. As of 2016, EPAS is entirely running in Windows environment on Physical servers which are located in Barclays' data centers.

Cloud technology has been out for years, and has already been adopted by numerous institutions and corporations. We believe cloud has reached its maturity to fulfill Barclays' needs, since it offers much more, and are more cost effective than physical servers. The team of WPI students carried out this cloud migration road map through following steps:

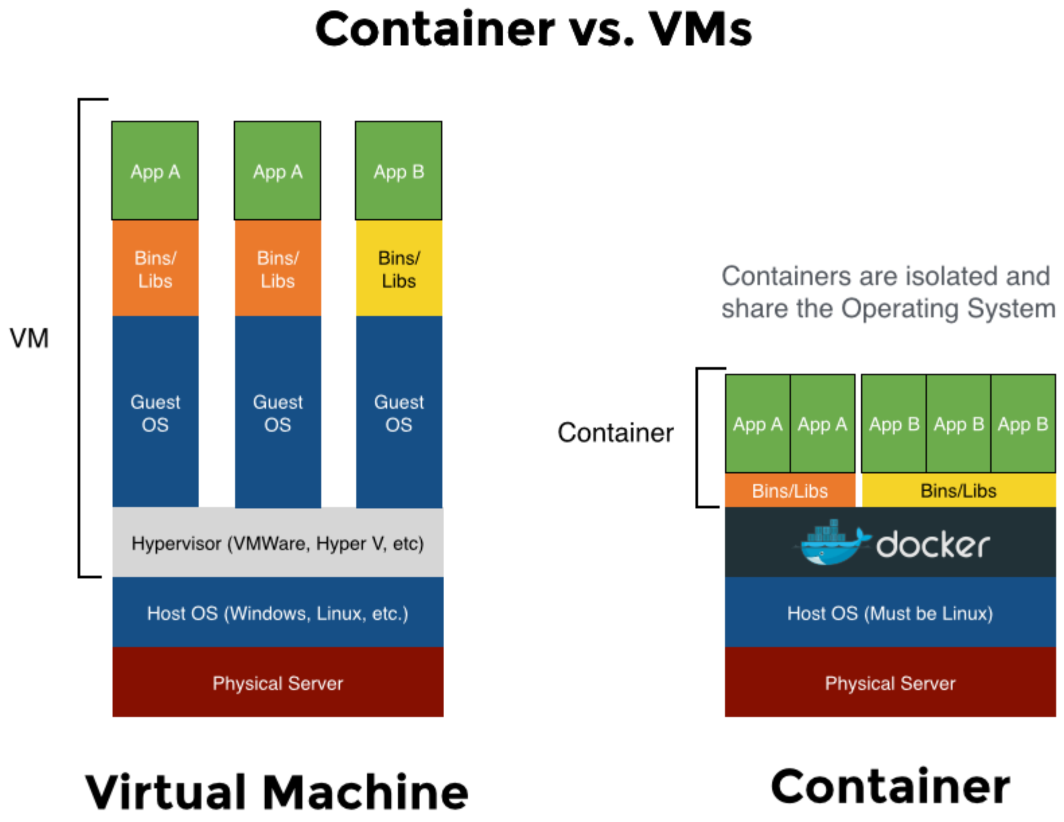
1. Investigate and test existing tools and technologies on the market which can aid our cloud migration process.
2. Create and set up the cloud environment using the tools investigated, and verify the capabilities and limitations of the cloud environment.
3. Migrate a minimal application onto the cloud, and test it under our cloud environment configuration.
4. Describe future works which need to be done, and concerns which need to be addressed for people who will continue this migration process after we leave in December.

Step-by-step guide

I. Technology Research

1. Containerization vs. Virtualization

Containers have similar resource isolation and allocation benefits as virtual machines but a different architectural approach allows them to be more portable and efficient.



https://serversideup.net/wp-content/uploads/2014/10/ContainerVsVM.png

2. Docker, Vagrant, Chef, Mono, Apache

Tool	Description	Official Site
Docker	Platform to pack, ship, and run any application as a lightweight container.	https://www.docker.com/

Vagrant	Tool for building and managing virtual development environment.	https://www.vagrantup.com/
Chef	Powerful automation platform that transforms infrastructure into code.	https://www.chef.io/
Mono	Open source development platform enables .NET framework applications to run on non-Windows environment.	http://www.mono-project.com/
Apache HTTP Server	Web server utilizes mod_mono as a middleware to run and host ASP.NET application across platform	http://httpd.apache.org/

II. Cloud Environment Configuration

1. Virtual Machine Creation (for internal offering)

In this road map, we're using Vagrant as the only tool to set up virtual machine, since it's relatively simple and straightforward. If you decide to use Chef, please refer to “IB Chef Development Environment” for instructions on setting up Test Kitchen** with Chef.

** Test Kitchen is a virtual machine provisioned by Chef and created with the help of Vagrant.

1) **Pre-requisites**

- If you're running from Windows machine, please make sure to have PuTTY installed.
- Make sure you already have a SI Infrastructure account.
- Make sure you have access to the KVM/libvirt hypervisors.

2) **Logon to your assigned hypervisor using your SI ssh key as yourself. For example,**

In Linux, open bash terminal:

```
ssh -A -i ~/.ssh/id_rsa xcrlxsie04d
```

In Windows, open PuTTY:

- Under the "Connection/SSH/Auth" tab, add your own private key path.
- Next, under the "Session" tab, please provide the host name, then click "Open"
- Then a console window will pop up, please provide your SI Logon User credential. After this, you'll be grant access to the hypervisor.

3) **Initialize a new virtual machine with Vagrant**

In the console, do:

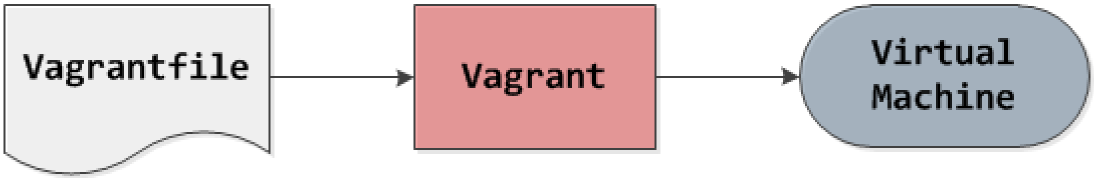
```
mkdir your-directory-name
cd your-directory-name
vagrant init       # Placed a Vagrantfile in the current directory
vi Vagrantfile     # Edit the Vagrantfile
```

In your Vagrantfile, replace the existing configuration code with the following:

```
Vagrant.configure(2) do |config|
  config.vm.box = "your-vm-box-name"
  config.vm.box_url = "http://your.vagrant.box.url"
  config.vm.communicator = "ssh"
    config.ssh.username = "root"
    config.ssh.private_key_path = "~your-username/.ssh/vagrant-pem"
    config.ssh.pty = "true"
config.vm.network(:private_network, {:type=>"dhcp", :libvirt__forward_mode=>"none"})
  config.vm.network(:forwarded_port, {:host_ip=>"x.x.x.x", :guest=>80, :host=>18111})   # You can pick a
host port from 18100 to 18200
  config.vm.synced_folder "/path/to/your/host/shared/folder", "/path/to/your/guest/shared/folder"
end
```

This script will configure the following for the new VM:

- Pull a vagrant box (image) from Barclays' internal storage
- Set up logon ssh authentication, using Vagrant's default ssh key
- Set up network type of the guest
- Set up forwarded ports and shared folders between host and guest



4) **Bring up the Vagrant VM**

```
vagrant up
```

5) **Login to the newly created VM**

```
vagrant ssh
```

---

6) **Can also destroy the VM**

```
vagrant destroy
```

---

7) **Inside the VM, host a simple HTTP Server using Python to make sure forwarded ports work correctly**

```
python -m SimpleHTTPServer <shared-port-number>
```

---

2. Container Creation Process

The Virtual Machine we were able to create is based on Barclays' internal image, which is running Red Hat Enterprise 6. Therefore, the following steps only intents to work on RHEL6. If you are working on other Linux versions or releases, please refer to Docker's official documentations at https://docs.docker.com/.

1) **Install Docker**

```
yum -y install docker-io    # install docker-io rpm package
yum -y update docker-io     # update the docker-io
```

The docker-io package includes docker and docker daemon. The Docker daemon is the persistent process that manages containers, and is required to run prior to the using of docker. To start the docker daemon, do:

```
service docker start
```

Can also use this command to start docker daemon at the start-up of the machine:

```
chkconfig docker on
```

---

2) **Write automation script in the Dockerfile**
We'll need to install mono and its modules, including xsp, mod_mono, and libgdiplus. Since Barclays' internal repositories don't offer these packages. We have to download and build them from source. To download:

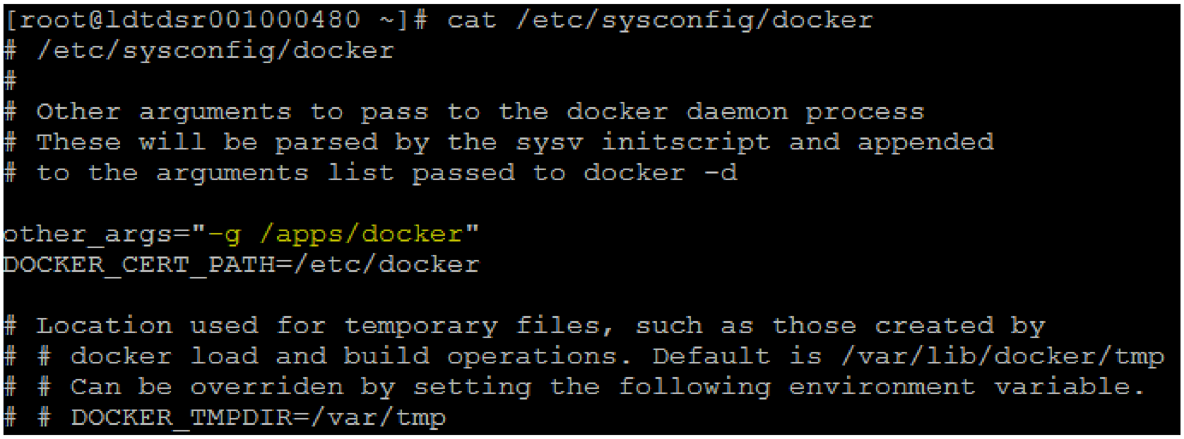
Tool	Version	File Name	Download Link
mono	4.6.2.7	mono-4.6.2.7.tar.bz2	http://download.mono-project.com/sources/mono/mono-4.6.2.7.tar.bz2
xsp	4.2	xsp-4.2.tar.gz	http://download.mono-project.com/sources/xsp/xsp-4.2.tar.gz
mod_mono	3.12	mod_mono-3.12.tar.gz	http://download.mono-project.com/sources/mod_mono/mod_mono-3.12.tar.gz
libgdiplus	4.2	libgdiplus-4.2.tar.gz	http://download.mono-project.com/sources/libgdiplus/libgdiplus-4.2.tar.gz

Append the following scripts to your Dockerfile:

```
# Add the source packages to the container
ADD mono-4.6.1.5.tar.bz2 /mono-src
ADD xsp-4.2.tar.gz /mono-src
ADD mod_mono-3.12.tar.gz /mono-src
ADD libgdiplus-4.2.tar.gz /mono-src

# Install dependencies for mono and its modules
RUN yum -y install gcc gcc-c++ libtool bison autoconf automake gettext httpd httpd-devel make unzip glib2-
devel libX11-devel pixman-devel fontconfig-devel freetype-devel libexif-devel libjpeg-devel libtiff-devel
libpng-devel giflib-devel cairo-devel

# Clear the previous installation cache
RUN yum -y clean all

# Compile and install libgdiplus first
RUN cd /mono-src/libgdiplus-4.2; ./configure --prefix=/usr; make && make install

# Then, compile and install mono
```

```
RUN cd /mono-src/mono-4.6.1; ./configure --prefix=/usr -with-libgdiplus=/usr; make && make install

# Add Package Configuration Path environment variable
ENV PKG_CONFIG_PATH /usr/lib/pkgconfig
RUN echo "export PKG_CONFIG_PATH=/usr/lib/pkgconfig" >> ~/.bash_profile

# Compile and install xsp and mod_mono
RUN cd /mono-src/xsp-4.2; ./configure --prefix=/usr; make && make install
RUN cd /mono-src/mod_mono-3.12; ./configure --prefix=/usr --with-mono-prefix=/usr; make && make install
```

Make sure that there are enough disk spaces on the virtual machine. Use the ‘df -h’ command to check that. In default, docker stores all its files in the ‘/var’ partition. Verify there’s at least 8GB free disk space in ‘/var’. If not, please consider change the docker’s base directory to a larger partition, since docker will report undefined error, if there isn't enough disk space. edit the file "/etc/sysconfig/docker", by:



```
[root@ldtdsr001000480 ~]# cat /etc/sysconfig/docker
# /etc/sysconfig/docker
#
# Other arguments to pass to the docker daemon process
# These will be parsed by the sysv initscript and appended
# to the arguments list passed to docker -d

other_args="-g /apps/docker"
DOCKER_CERT_PATH=/etc/docker

# Location used for temporary files, such as those created by
# # docker load and build operations. Default is /var/lib/docker/tmp
# # Can be overriden by setting the following environment variable.
# # DOCKER_TMPDIR=/var/tmp
```

3) **Build a docker image with the script**

We can build the image with our Dockerfile. We do:

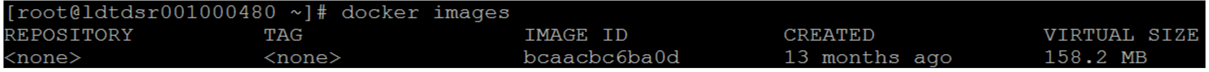
```
docker build -t "pick-a-name:version" .
```

This building process is going to take roughly one and a half hour. After it's done. The docker image is ready, you can see by checking "docker images".

Check the image by:

```
docker images
```

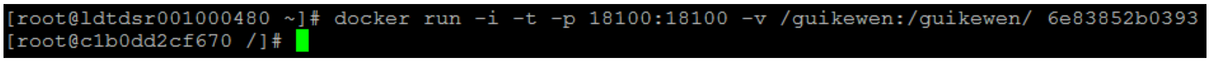
The output should look like:



```
[root@ldtdsr001000480 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
<none>              <none>              bcaacbc6ba0d        13 months ago       158.2 MB
```

Then, we can create and run a container based on the image we created:

```
docker run -i -t -p 18100:18100 -v /guikewen:/guikewen/ 6e83852b0393
```



```
[root@ldtdsr001000480 ~]# docker run -i -t -p 18100:18100 -v /guikewen:/guikewen/ 6e83852b0393
[root@c1b0dd2cf670 /]#
```

Some points:

- -i, interactive, keep STDIN open even if not attached to the container
- -t, allocate a pseudo-TTY
- -p, publish a container's port to the host
- -v, bind mount a volume (e.g., a shared folder)

III. Minimal Application Implementation

- SpaceManager
    - First, build it on Windows
    - Second, transfer the "Debug" folder through WinSCP to the hypervisor → virtual machine → container.
    - Then, run the SpaceManager inside the container by:

      ```
      # cd /your/path/to/Debug
      mono-service SpaceManager.exe
      ```

IV. Future Implementation Recommendation

- Investigate further on Chef and Mono’s offerings
    - Chef
        - https://docs.chef.io/
        - IB Chef Development Environment
    - Mono
        - http://www.mono-project.com/docs/

- Consider reconfiguring the existing authentication mechanism to adopt SSL or Kerberos
    - https://www.ibm.com/support/knowledgecenter/SSMKHH_9.0.0/com.ibm.etools.mft.doc/ap12240_.htm

    - http://www.easysoft.com/products/data_access/odbc-sql-server-driver/kerberos.html

- Consider using Kubernetes or Docker Swarm for deploying, scaling, and managing large amount of container instances
    - http://kubernetes.io/docs/
    - https://www.docker.com/products/docker-swarm
    - https://www.docker.com/cp/docker-and-kubernetes

- Gradual migration of EPAS's modules

V. Acknowledgements

Martin East, Anu Koolothumkandi, and the SI-Engineering Team

Barclays Cloud Team