Thunder Cloud

by

Alexious Sherman, Cameron Varker, Sean Ward

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Alexious Sherman
Alexious Sherman – Project

Cameron Varker
Cameron Varker – Network Specialist

Sean Ward
Sean Ward – Software Developer

April 5th, 2019
Date

Ryan Moore
Ryan Moore – Advisor

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2019

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

TABLES

FIGURES

ABSTRACT

Amazon Web Services (AWS) and Microsoft Azure are the two leading cloud solution services in a market worth $18.2 billion in 2016, and it is expected to worth $40 billion by the end of 2018. Microsoft's Azure market share has grown from 8.7% in 2016 to 29.4% in 2018 thanks, in large part, to their cloud-first strategy in recent years. Many enterprises have been left cloud-locked, leaving them stuck working with either AWS or Azure due to budget and time restrictions, while some larger enterprises utilize both AWS and Azure. Being able to securely and cheaply switch between cloud services today will save companies time and money, as well as keep their options open if one works better for them than the other. Thunder Cloud will help solve this cloud lock issue between AWS and Azure for companies by taking data from one service, converting it the format of the other, and then uploading it to that platform.

1. Introduction

1.1 Problem

Amazon Web Services (AWS) and Microsoft Azure are the two leading cloud solution services in a market worth $23.6 billion in 2017, which was up from $18.2 billion in 2016, and it is expected to worth $40 billion by the end of 2018. This rapidly growing market for cloud solutions has led to many wanting to switch between solutions or even use both for various reasons. Azure's market share has grown from 8.7% in 2016 to 29.4% in 2018. As a result, there are a significant number of users and enterprises switching from AWS to Azure or looking to utilize both. Some of these entities are experiencing cloud-lock which is when an enterprise is locked into a single cloud service due to the significant time and costs to switch from their current platform. Switching is a difficult prospect as AWS and Azure use proprietary storage which causes issues when converting between them including data compatibility, access control, and other security issues. Thunder Cloud, our conversion tool, will simplify and securely transfer data between AWS and Azure.

1.2 Solution

The Infrastructure as a Service (IaaS) market is divided mostly between AWS and Azure. This has created a situation where many enterprises and users will attempt to use both or convert between one or the other. Both AWS and Azure use proprietary forms of data and follow no standard which has complicated conversion processes. There have been several industry attempts to fix this problem, such as an entire page on Microsoft's website dedicated to a guide on

converting from AWS to Azure. This guide and others are not always easy, quick, and/ or secure solutions. The Microsoft guide, for example, is a lengthy process which involves attempting to recover an AWS image through Azure's recovery process. However, as of today there is no real solution to cloud-lock.

Our solution will be to write a universal tool that can translate AWS data to Azure data. Our tool will pull the data of the environment down from one IaaS, Azure or AWS in this case, provider and convert it to the type of the other IaaS. The conversion will be an automated script/piece of software which will automatically perform steps and edit data to make it compatible with the other IaaS environment. After the conversion, the service will push the data back out to the opposite IaaS. This tool will look to cut down the time it takes for an entity to convert a machine from one cloud service to another; thus, this process will reduce the impact of cloud-lock on an entity.

1.3 Project Goals

This report will discuss the origination of our project in a discussion after class with Bogdan Vykhovanyuk and Tyler Hopperton. Further, we will discuss our methods and requirements and how they evolved over time. The initial requirements were too broad and made the scope too much to bear, so we tightened up our requirements. Our methodology includes using scripts and object-oriented code to automate a process of downloading, converting, and uploading VMs and databases from AWS to Azure.

1.4 Overview

The remainder of this final report outlines, in detail, how the project was completed. The report includes the following sections: design objectives, methodology, budget, timeline, problems encountered, and future recommendations.

2. Discussion

2.1 Project Concept

The project was inspired from a conversation with advisors Bogdan Vykhovanyuk, Tyler Hopperton, and our team. It was suggested that a common issue today was cloud-lock, especially between Azure and AWS. Our team took this problem and researched it to its extent to create a solution to resolve the problem.

Our research led us to believing that it would be possible to convert AWS and Azure data and automate that process. The process will allow users to quickly, cheaply, and safely move data. We accomplish this by downloading data from one IaaS converting it and uploading it up to another IaaS on a local machine.

2.2 Design Objectives

We plan to deliver a tool that will convert data from AWS to Azure and back. Originally, our scope included any IaaS and all types of data. This scope was far too broad, and our initial goal needed to be tightened up to allow us to deliver a quality project. We shed the idea of doing any IaaS and converting anything, and instead we have decided to try to convert VMs and databases from AWS to Azure and vice versa. *Table 1: Design Objectives* below show the requirements we had for Thunder Cloud.

Requirement	Description	Priority
Quick and Simple	The tool will require less steps for the user and less time than other methods found on the internet	High
Data Down	The tool will be able to automatically pull data down from AWS or Azure	Medium

Conversion	The tool will be able to take data from AWS or Azure and convert it to the format of the other	High
Data Up	The data will automatically get pushed back to either Azure or AWS after it is converted from the other.	Medium
User Interface	The user interface will take a source and destination string and contain a button which will call the process.	Low

Table 1: Design Objectives

2.3 Technical Approach

For this semester we decided on using a single file format for converting machines from one cloud service to another, the VHD (Virtual Hard Drive). A VHD file can be produced and consumed by both Amazon and Azure which lends itself to be a good file format to keep consistency. To achieve conversion, we use Amazon Web Services' Command Line Interface and Microsoft Azure's Command Line Interface (CLI). For each cloud service we have set up user roles and policies to allow imports and exports of virtual machines. Specifically, each cloud service has a set of commands from their command line interface that allows conversion, uploading, and prepping for imports and exports. After completing manual ability through Azure and AWS CLIs, we stood up a Spring Boot web service written in Kotlin. This web service has 4 endpoints to import and export Azure and AWS which can accessed with HTTP (Hyper Text Transfer Protocol) web calls using the verb GET. Once the endpoint is called upon the endpoint calls on the service which executes batch scripts using the AWS and Azure CLI commands. This batch script is called upon through the Java library Process Builder. Once the script executes all CLI commands happen sequentially then return the output of the command prompt.

2.4 User Profile

Table 2: User Profile as shown below shows the kind of people we had in mind when we were designing and developing Thunder Cloud. Throughout the project, we referred back to this table whenever we were looking to make design changes or adding additional features to Thundercloud.

User Profile
PROJECT: Thunder Cloud: A cloud migration tool.
POTENTIAL USERS: - Cloud Administrators or someone with a similar company job - Technically inclined individuals interested in migrating from AWS to Azure or vice versa.
SOFTWARE, INTERFACE, AND RELATED EXPERIENCE: Our project will primarily be targeted at companies that use AWS and Azure or want to convert over from one to the other. More specifically it will be targeted at the highly skilled individuals at those companies who work with Azure and AWS. Individuals will be able to the use the tool as well, but they are less likely to need such a tool.
EXPERIENCE WITH SIMILAR APPLICATIONS: - Amazon Web Services (AWS) - Microsoft Azure Currently we are unsure as to what the UI is going to look like and there aren't good applications out there for cloud migration so there are not any applications that the user could be familiar with that would help them use our tool.

TASK EXPERIENCE: - Experience navigating Azure and AWS - Experience navigating applications and imputing data or a path to data
FREQUENCY OF USE: For some users the application could be used as frequently as weekly or daily if they use both AWS and Azure but could be as little as a one-time use if they are just converting from one to the other
KEY PROJECT DESIGN REQUIREMENTS THAT THE PROFILE SUGGESTS: - Convenient interface - Automated transfer from AWS to Azure or vice versa

Table 2: User Profile

2.5 Use Case Diagram

Figure 1: User Case Diagram presents the use case diagram. As of now this diagram represents the requirements needed for Thunder Cloud. We may add more throughout the semester.

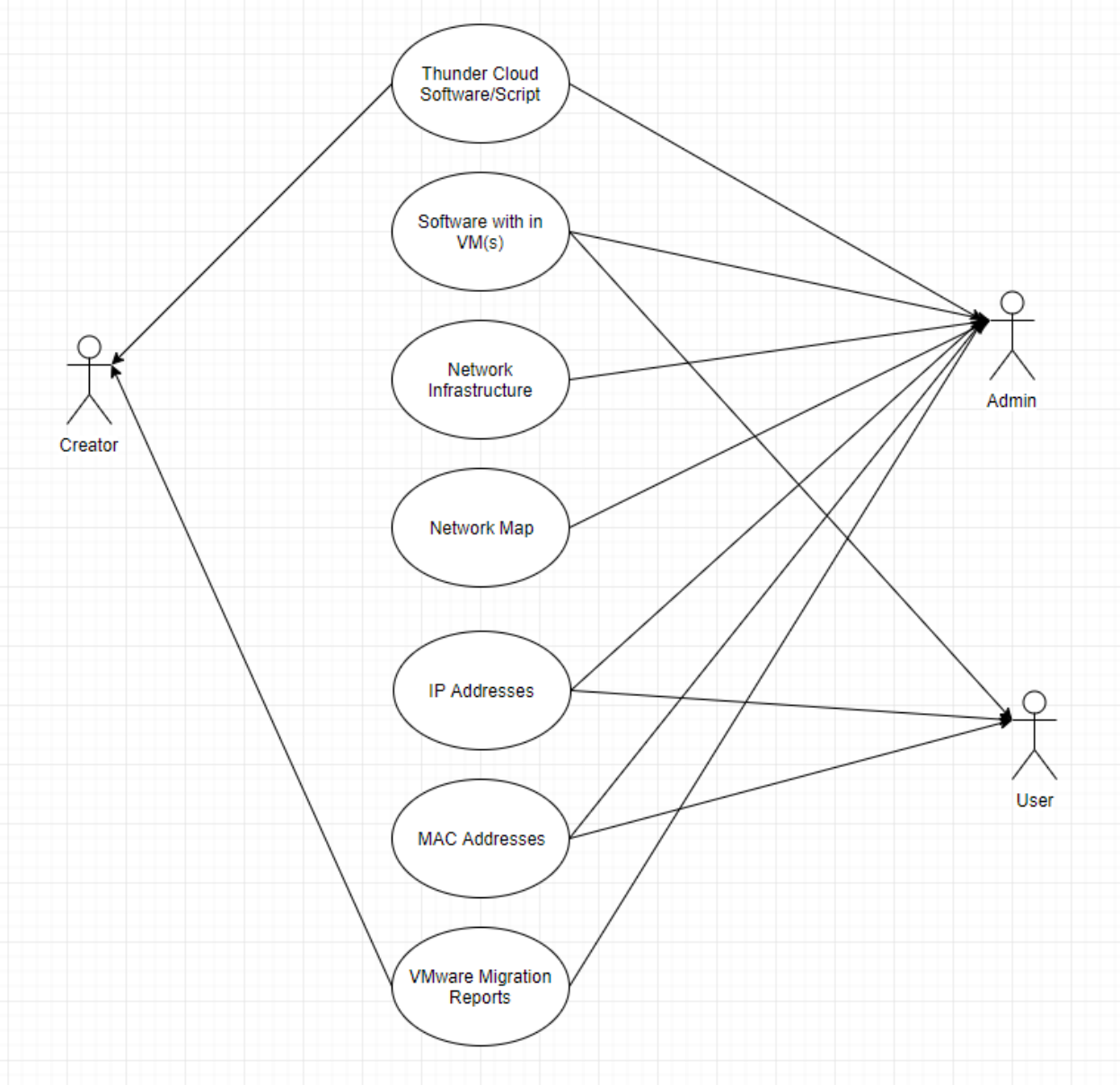

Figure 1: Use Case Diagram

2.6 Technical Architecture

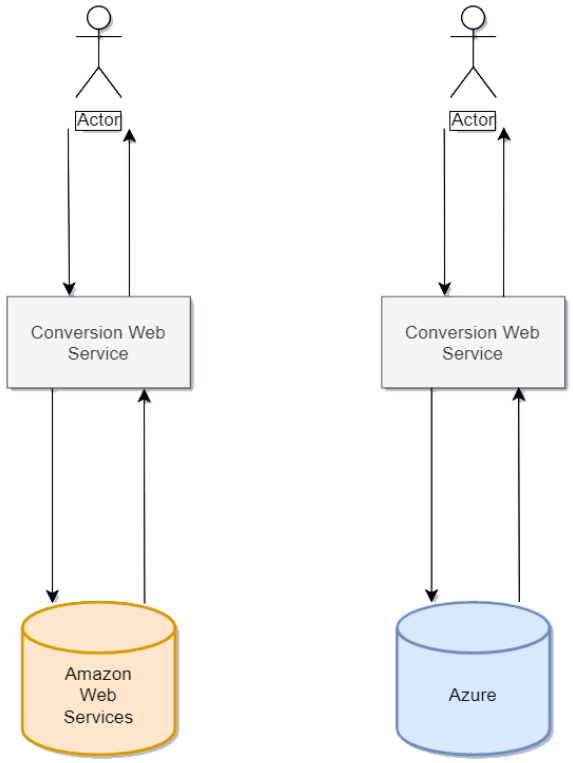

Figure 2: Technical Architecture Diagram

2.7 Technical Discussion

The user puts in request in our web service. The webservice takes these requests and sends them to the appropriate cloud services command line interface based on where it is being exported. Through the command line, our service prepares the VM for export and exports it. It will than throw back any errors it receives or proceed to switching to the other cloud services' command line to run commands to upload the VM. The web service will then return a statement saying that the process was executed successfully. This web service exports and imports data as VHD files. REST endpoints communicate in JSON. the Service runs CMD and executes Command Line Interface (CLI) programs.

2.8 Testing

The Required Elements are:

1. Overview/Methodology (include why you chose the approach you did)

Overview

The primary mode of testing used will be Unit Testing, an automated way to test individual pieces of code. Our automated scripts are controlled by a web service which needs to be tested in this well accepted way to automate testing of code.

2. Scope of Testing (what will be covered and why)

Our scope for testing is to convert Linux machines as well as windows machines to make sure that our apps works for our target users. The test will be broken down into two parts: the conversion of a Linux back and forth automatically and then the conversion of a windows machine back and forth automatically.

3. Objectives

The objective of our testing is to make sure people can successfully convert a Linux or Windows machine from Azure to AWS or vice versa. Each of the two parts will have 3 phases. Phase one will be setting up the virtual machine for transport and setting up the script. Phase two will be running be running the ThunderCloud service. Phase 3 will be testing the virtual machines once they are transferred to make sure they work the same as they did before. This will include checking for test files to make sure they are still in the same location.

4. Logging Test and Procedures (including the results of your testing and Pass/Fail Conditions)
Unit Testing:

Test Case	Import Export	Section	Purpose	Expected Outcome	Pass Fail
1	Import	Azure Controller	This test will assess whether the controller can receive a payload of data for importing Azure and successfully call the Azure import service.	Pass	Pass
2	Export	Azure Controller	This test will assess whether the controller can receive a payload of data for exporting Azure and successfully call the Azure export service.	Pass	Pass
3	Import	AWS Controller	This test will assess whether the controller can receive a payload of data for importing AWS and successfully call the AWS import service.	Pass	Pass
4	Export	AWS Controller	This test will assess whether the controller can receive a payload of data for exporting AWS and successfully call the AWS export service.	Pass	Pass
5	Import	Azure Service	This test will assess whether the service can successfully call the shell script for importing Azure.	Pass	Pass
6	Export	Azure Service	This test will assess whether the service can successfully call the shell script for exporting Azure.	Pass	Pass
7	Import	AWS Service	This test will assess whether the service can successfully call the shell script for importing AWS.	Pass	Pass

8	Export	AWS Service	This test will assess whether the service can successfully call the shell script for exporting AWS.	Pass	Pass
9	Import	Azure Controller	This test will assess whether the controller will return a 400 HTTP error when given incorrect, badly formatted, or otherwise bad data.	Pass	Pass
10	Export	Azure Controller	This test will assess whether the controller will return a 400 HTTP error when given incorrect, badly formatted, or otherwise bad data.	Pass	Pass
11	Import	AWS Controller	This test will assess whether the controller will return a 400 HTTP error when given incorrect, badly formatted, or otherwise bad data.	Pass	Pass
12	Export	AWS Controller	This test will assess whether the controller will return a 400 HTTP error when given incorrect, badly formatted, or otherwise bad data.	Pass	Pass
13	Import	Azure Service	This test will assess whether the service will throw an exception when given the shell script throws an error.	Pass	Pass
14	Export	Azure Service	This test will assess whether the service will throw an exception when given the shell script throws an error.	Pass	Pass
15	Import	AWS Service	This test will assess whether the service will throw an exception when given the shell script throws an error.	Pass	Pass

16	Export	AWS Service	This test will assess whether the service will throw an exception when given the shell script throws an error.	Pass	Pass

Table 3: Unit Testing Table

When we go to log and test, we will have each person testing Thundercloud note any issues or bugs they ran into as well as any questions they had or still have so that we can make sure they are fixed or answered in support documents. We will attempt to fix any bugs we run into along the way if they stop progress, otherwise we will compile them after testing has been run once and fix them as needed.

2.9 Budget

Below, Table 4: Spreadsheet, is a picture of our budget for our project as if we worked full time for a large enterprise. We planned for each of our two cloud admins to be working 80 hours each over two months and our developer to be working 160 hours over 2 months. We based our labor costs on what jobs in the market are paying these kinds of employees are making per hour and then multiplied that by the numbers hours they worked. These full-time employees are representative of ourselves if we were working for a company full time working on this project. In reality, this project cost next to nothing to do as we mostly used Amazon's and Azure's free tier options. We did however spend roughly $60 on AWS support and building some VMs in Azure.

No.	Item	Unit Count	Unit Price	Line Item Total
Labor				
1	Developer	160 Hrs.	$36	$5,760
2	Cloud Admin	160 Hrs.	$43	$6,880
Cloud Support				
3	AWS	2 Months	$100	$200
4	AZURE	2 Months	$100	$200
		Subtotal		**$13,040**
		Team Total		**$0**

Table 4: Budget Spreadsheet

2.10 Gantt Chart

Table 3: Gantt Chart Spreadsheet and *Figure 2: Gantt Chart Figure* is a schedule of our project for the fall and spring of the 2018-2019 school year.

TASK	START DATE	END DATE	DURATION (Days)
Research	**8/28/2018**	**3/11/2019**	**195**
Team Members & Project Names	8/28/2018	9/3/2018	6
Team Draft Contract	9/4/2018	9/26/2018	22
Planning	**9/26/2018**	**1/16/2019**	**112**
Abstract & Team Final Contract	10/7/2018	10/15/2018	8
Environment Learning AZURE/AWS	10/16/2018	10/25/2018	9
User Profile & Use Case Diagram	10/16/2018	10/22/2018	6
Local-2-Cloud Migration	10/20/2018	10/30/2018	10
Draft Report	10/23/2018	11/5/2018	13
Cloud-2-Cloud Migration	10/30/2018	11/6/2018	7
Migration Method Experimenting	10/30/2018	11/10/2018	11
Final Report	11/6/2018	11/26/2018	20
Development	**11/9/2018**	**2/22/2019**	**105**
VM Set-Up	11/9/2018	11/10/2018	1
Network Infrastructure Development	11/9/2018	11/13/2018	4
Tool Development	11/9/2018	2/2/2019	85
User Interface Development	11/9/2018	2/2/2019	85
Automating tactics	11/10/2018	11/20/2018	10

Automation Experimenting	11/10/2018	11/25/2018	15
Presentation	11/27/2018	12/3/2018	6
Deployment	2/2/2019	2/22/2019	20
Testing	**2/22/2019**	**3/11/2019**	**17**
Review	**3/11/2019**	**3/31/2019**	**20**

Table 5: Gantt Chart Spreadsheet

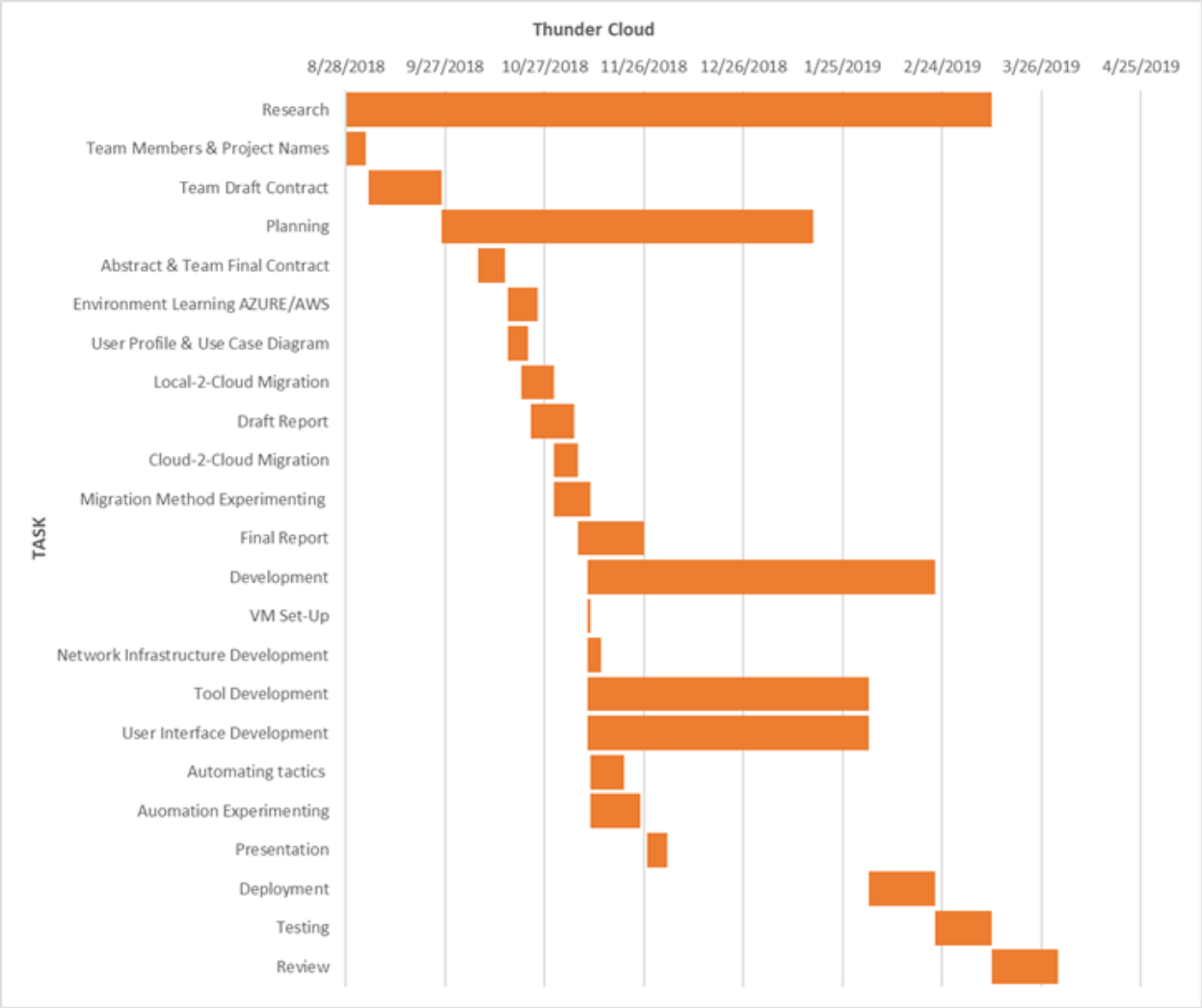

Figure 3: Gantt Chart Figure

2.11 Problems Encountered

At the beginning of the semester, we were fairly unfamiliar with AWS and Azure for anything other than setting up VMs. We have learned a lot so far but have encountered many issues as shown in *Table 6: Problems* below.

Problem	Resolution
Original Scope Was Too Large	Scope was tightened up to include only AWS and Azure, VMs and databases.
Amazon limits what people can people export	Further research during the spring semester
AWS and Azure both have their own commands	Spend time researching and learning how to use both command lines
Having trouble generalizing Unix based VMs in Azure	Going to meet with advisor and see if he can guide us in the right direct as well as talking to industry experts

Table 6: Problems

2.12 Future Recommendation

If the project could be started from the beginning again, it would be best to work this project as a web service first and the solution to the problem second. This way more time could have been spent exploring the options of how to interact with AWS and Azure APIs (Application Programming Interface). Learning the Cloud took a major portion of our time in the Fall which left less time to stand up the web service properly and reduced the time our programmer could teach the networking students how the web service worked.

If the team had more time it would be prudent to expand the web service to have a front end created as well as expanding into more cloud solutions such as Google Cloud. Further, support for other VM solutions, which are not cloud based, such as a VMware. This would open the service to being a central hub for all VM transformations. More technical solutions such as leveraging AWS' and Azure's APIs and expanding to more devices and entire networks.

Suggestions from others have included utilizing AWS' and Azure's APIs and expanding into more cloud solutions. These were known going and cut from the scope to focus on Azure and AWS, the largest cloud providers.

3. Conclusion

3.1 Fall Semester

We are still in our research and planning phase of our design. Our group at the beginning of the semester didn't fully understand just how much research we would have to do for this project. We are now realizing that our project is going to be way more researched based and that once we figure out what need to do, it seems like the development part will be pretty short. Based on our Gantt chart, we are currently behind as we have not done much development yet. However, as a group we feel that we are not behind and that we just miscalculated our Gantt Chart because of a lack of knowledge on what our project actually required.

We are currently in the process of understanding how AWS and Azure handle importing and exporting each virtual machine. Both AWS and Azure have their one command lines that will be used to generalize virtual machines which is necessary in order to export the VMs which we are currently trying to learn. Through research we are finding that AWS is very limiting in what it will let us do, so we have spent the latter part of the semester working exclusively with Azure to ensure we had something to show off during our end of semester presentation.

3.2 Spring Semester 2019

The team expanded their understanding of Spring Boot, Kotlin, and web services greatly by creating a Kotlin using the Spring Boot Framework. We expanded our experience with web services and leveraging the power of software to provide functionality in an easy to use manner. For both the team's developer and the team's networking specialists there was a lot to learn from

both sides about clouds as well. It was a fantastic opportunity to learn about this emerging technology that is pivotal to all areas of technology today.

Expo taught the team more about how to pitch and present to professionals in the field rather than a room full of students and faculty. It was a great experience to be exposed to these people and network with them for job opportunities. Expo was an important step in all team member's careers outside of this class.

Appendix A: Additional Information

Appendix B: References

Central, A. (2018, May 03). PRESS RELEASES. Retrieved from https://www.asicentral.com/news/press/press-releases/january-2018/asi-reports-2017-distributor-sales-of-promo-products-hit-record-236-billion/

Coles, Cameron. "AWS vs Azure vs Google Cloud Market Share 2018 Report." *Skyhigh*, Skyhigh Networks, 12 Sept. 2018, www.skyhighnetworks.com/cloud-security-blog/microsoft-azure-closes-iaas-adoption-gap-with-amazon-aws/

Azure vs. AWS - Which Cloud Service is Right for You? (2017, November 29). Retrieved from https://www.onlc.com/blog/azure-vs-aws-cloud-service-right/