Designing a Modern Software Engineering Training Program with Cloud Computing

A Major Qualifying Project report:

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfilment of the requirements for the degree of

Bachelor of Science

by

Brett Cohen

Brianna Greenlaw

Date:

Approved:

Professor George Heineman, Major Advisor

Table of Contents

Abstract

The software engineering industry is trending towards cloud computing. For our project, we assessed the various tools and practices used in modern software development. The main goals of this project were to create a reference model for developing cloud-based applications, to program a functional cloud-based prototype, and to develop an accompanying training manual. These materials will be incorporated into the software engineering courses at WPI, namely CS 3733 and CS 509.

Executive Summary

Cloud computing is a model for enabling on-demand, convenient, and ubiquitous access to shared computing resources. Industry leaders are looking into cloud computing to increase efficiency in products and the workplace.

Our project involves researching the current state of the software industry, including desired skills, emerging technology, and other trends as of October, 2017. We will use this data to compile the necessary skillset of a full stack cloud developer. Our findings will be incorporated into the Software Engineering courses at WPI, namely CS 509 Design of Software Systems and CS 3733 Software Engineering.

The main categories we researched were cloud computing itself, cloud developer skillsets, prevalent technologies, industry initiatives, and cybersecurity. We analyzed data from a variety of job postings related to cloud computing. We interviewed various company professionals about the skills they expect for cloud computing positions.

We used the data collected from the above methods to create three key components of the project. These components are a reference model, a cloud-based application, and a training manual.

The reference model is a tiered grouping of technologies or methodologies that are required to implement an application similar to the prototype application developed for this project. The tiers are ordered by importance to development. We present options for each tier that are derived from the research in this paper.

The cloud-based application is an image classification service that allows users to upload, tag, and search for images. The application was run entirely on the cloud and was built using the reference model previously mentioned.

The training manual consists of modules that aim to guide a novice cloud developer through their first cloud application. The modules can be completed independently, but together form a guide that teaches the basics of cloud development using Amazon Web Services (AWS). In the end, the developer will become familiar with several services offered by AWS and should be able to build an image classification service similar to our own.

We end this report by discussing our technology choices and alternatives, as well as limitations of the project.

1. Introduction

Cloud Computing is an emerging field of Software Engineering. The National Institute of Standards and Technology (NIST) defines cloud computing as:

> Cloud computing is 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. (NIST, 2011)

Cloud Computing can be broken into five essential characteristics, three service models, and four deployment methods. All of these will be discussed more in-depth in Section 2.1 of the background.

Industry leaders are looking into cloud computing to increase efficiency in products and the workplace. Ford Motor Company, Coca Cola, and JDA Software, among many others have moved their products and services over to cloud computing. Currently the top three leaders in cloud distribution are; Amazon, Microsoft, and Google. According to Forbes, the forecast for 2018 of the percentage of Information Technology (IT) spending directed at cloud computing resources and technologies is going to reach 60% (Columbus, 2017).

Cloud computing allows for companies to move away from having designated IT specialists for downloading and maintaining software on their devices by making the software more easily available using technologies such as web services and web applications. With access to a cloud platform, companies do not have to spend money to store their data and software on-site, or in an expensive, dedicated off-site location. Companies can also increase their data storage without having to build their own data storage center. They simply pay the cloud providers for more storage. Companies are also able to downsize their data storage amounts if business rules for storage of data change, or the company downsizes their products.

Our project involves researching the current state of the software industry, including desired skills, emerging technology, and other trends. We will use this data to compile the necessary skillset of a full stack cloud developer. Our findings will serve to capture the current state of the industry as of October, 2017. These findings will be incorporated into the Software Engineering courses at WPI, namely CS 509 Design of Software Systems and CS 3733 Software Engineering. Additionally, we will create an application that utilizes cloud technology as a proof of concept project for the course.

With the rise of popularity in cloud development, students must be prepared for a shift in which skills are required to succeed in the software industry. Currently, there is a lack of cloud

development principles and skills found in the curriculum. By incorporating these principles in the Software Engineering course, WPI will better prepare students with a more modernized skillset.

The goal of our project is to create a robust training structure for full stack cloud developers. Additionally, we will build an image classification service using cloud technology. To achieve this goal, we will achieve the following objectives:

1. Identify current requirements for cloud development positions in the industry
2. Identify initiatives by leading companies in the industry regarding cloud development
3. Research current and potential future cloud technologies, including microservices, databases, and DevOps
4. Research potential issues with cyber security

2. Background

2.1 Breakdown of Cloud Computing

Cloud computing is a model for enabling on-demand, convenient, and ubiquitous access to shared computing resources. Cloud computing can be broken into three large sections, with multiple subsections in each. The overarching sections are: essential characteristics of cloud computing, service models for use of the cloud by companies and individual users, and deployment methods for hosting the cloud. (NIST, 2011)

Cloud computing as a whole is supported by the Internet which allows end-users and service providers, such as Netflix, to use resources that might not be located on their local area network (LAN). What this means is that end-users are able to access data, such as movies in the case of Netflix, that could be located thousands of miles away. The primary benefit of cloud computing is that the resources or online services can be accessed over the network at a speed much faster than before (Microsoft, 2017). Both end-users and large companies using resources hosted on the cloud have reduced costs for varying reasons, both do not have to go out and purchase physical copies of the services; the companies do not have the extra cost of building and maintaining large infrastructure for data storage (Griffith, 2016); finally, companies are given the ability to see their usage of the services they pay for and can scale up or down depending on the demand, which reduces costs but also frees up the services so that other companies may use them (Amazon, 2017).

There are many different user types that will use cloud computing. The **end-user** installs programs offered, uses programs and websites that are hosted on the cloud, and stores data on the cloud such as emails, photos, and social media information. The **developers** that create services and software that is hosted on the cloud use the cloud to host their programs without the costly infrastructure setup and maintenance. The cloud platform company uses the cloud they created to make money. The **company** can do this by selling their cloud out to developers to host their services and charge based on the amount their service is used. The company could also charge end-users to use in-house developed cloud services such as data storage programs. Throughout this document, we refer to all of these types of users as "the user" of the cloud for simplicity.

2.1.1 Essential Characteristics

The essential characteristics of cloud computing can be divided into five categories that are in place to help both the provider and the consumer of the cloud service. The characteristics laid out by NIST are as follows, on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. All of these characteristics allow cloud computing to provide a seemingly unlimited supply of resources to users. (NIST, 2011)

On-demand Self-service

　　With cloud-based service, users do not have to interact with each service provider to complete projects that might need access to the server or need network storage. This allows users to remove the 'middle-man' aspect of traditional computing techniques to run more smoothly and efficiently. With non-cloud based technologies, users rely on other users and service providers to allow access and return results. (NIST, 2011)

Broad Network Access

　　Technology device types change constantly. Thus, access to the cloud with any type of device is mandatory. Broad network access allows for any type of device to utilize to the resources stored on the cloud. Types of devices that are included currently are mobile phones, tablets, desktop computers, and laptops. In the future devices such as cars could be added to the required device list. (NIST, 2011)

　　There are two major network types for connecting devices to the internet and each other. A Local Area Network (LAN) connects devices, by either Ethernet cable or wireless, to the same network of restricted size, most likely a home or office building. A Wide Area Network (WAN) connects devices in a wider scale than LAN, but not in a defined size. A WAN can be as small as a college campus or as large as a global network. (Meyers, 2012)

　　Broad network access is similar to WAN in that it allows connects from multiple types of devices from over large distances, but in a manner that only gives access to authorized users on the network. This security allows workers to access private, company-owned resources from devices at home, on business trips, and vacations using varying types of devices such as desktops and mobile phones. ("What is Broad", n.d.)

Resource Pooling

　　Currently, for non-cloud based technologies, most resources are located on the local machine, or in a data warehouse that an individual can access. With cloud computing, the resources are stored remotely on a server hosted by the cloud distributer Any user can now access these resources, with what is called the multi-tenant model. Resources that would be shared in this manner, and reassigned based on demand, include network bandwidth, storage, memory, and processing. (NIST, 2011)

　　With non-cloud based technologies, resources had to be located onsite or even on the local hard drive of the machines that required access to these resources. Alternatively, two users that wanted to share resources could create a group with the correct configurations that allowed file sharing across computers.This file sharing would be done using a shared drive stored locally or at an offsite location that was owned and operated by a company (Meyers, 2012). Resources not shared using file sharing had to be purchased individually and uploaded manually onto each device, which wasted time and money for companies and individual end-users.

　　With resource pooling, the file sharing aspect of allowing other users to have access to one another's files has been moved to the internet and not in a datacenter in an offsite location.

This allows for resources to be used by virtually anyone without having to wait for the resource to be retrieved from the center and can be used by multiple people without slowing the retrieval location down, thus making the resource scalable to large demands. ("What is Resource", n.d.)

Rapid Elasticity

With more and more users and companies moving their applications and services to the cloud, rapid elasticity helps reduce the impact of this large increase in demand. Rapid elasticity allows for the cloud's capabilities to be provisioned and released in a way that is flexible and reactive to demand. They are stretched in different lengths based on demand for the services. This capability allows users to have the view that resources are unlimited and can be taken at any time and in any amount. (NIST, 2011)

In non-cloud based technologies, when more space or resources are needed for the specific service or company to continue functioning as normal, large data centers need to be created which can take time and money. When a data center was needed, space needed to be rented or purchased, a building erected, and the infrastructure created and maintained. These requirements are removed for users once they move to the cloud (Meyers, 2012). Rapid elasticity allows users to access resources as needed, reducing time and cost. Furthermore, users do not need to build and configure their own resources, as it is handled automatically by the cloud provider ("What is Rapid", n.d.).

Measured Service

Cloud systems monitor the usage amount of each user account to optimize resources. By monitoring and charging for services, such as bandwidth and storage, companies can charge clients based on their individual usage. With this capability, resource usage is transparent for both the provider and the consumer. (NIST, 2011)

In non-cloud based technologies, users receive an invoice statement in which each charge is listed out at the end of a billing period. With measured service, users can monitor their usage before the billing period. Measured services also capture services being paid for but not being used so that users may reduce their plans accordingly. ("What is Measured", n.d.)

2.1.2 Service Models

The service model describes the three different ways that the cloud services can be utilized by consumers. The models can be broken down into: Software as a Service, Platform as a Service, and Infrastructure as a Service. These models allow consumers to have an understanding of the limitations of using the cloud for various purposes and for providers to keep consumers from overstepping their limitations depending on the type of service they are using. (NIST, 2011)
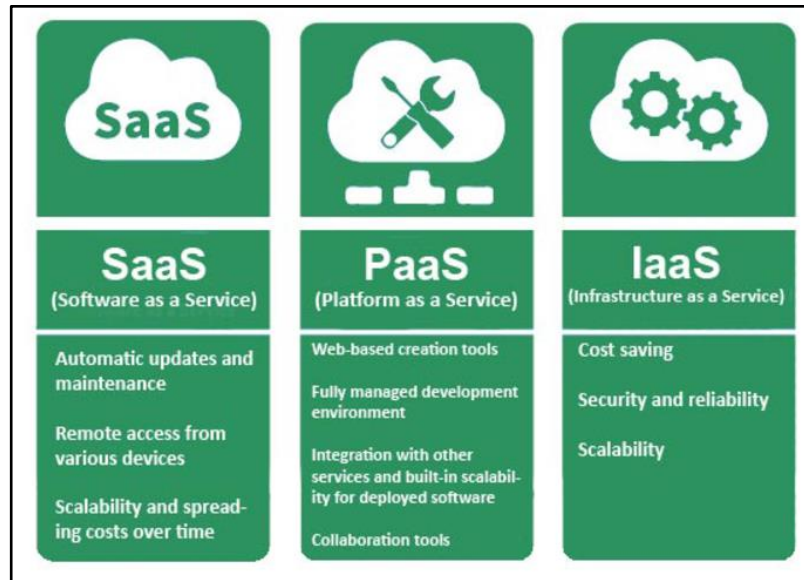
Figure 2.1. Offerings of each cloud service model. (Plitchenko, 2016)

The three service models offered by cloud computing can be broken down into the characteristics seen in Figure 2.1. The three model types have relatively little in common when it comes to the specific offers users gain from each. Based on research discussed below, the different service models have only one thing in common and that is that none of the users may modify the underlying cloud infrastructure of their specific provider's cloud.

Software as a Service (SaaS)

Software as a Service allows the user to gain access to the provider's applications that are running on a cloud infrastructure. The accessibility of the applications can be from different client devices through either a program interface or a thin client interface, such as a web browser. The user does not have the ability to change or manage the underlying cloud infrastructure, with the possible exception of limited user-specific configuration settings within the application. The user should not be able to change any of the following: network, servers, operating systems, storage, or even some individual application capabilities. (NIST, 2011)

Platform as a Service (PaaS)

Platform as a Service allows the user to deploy their own applications and services onto the cloud; these programs can be user-created or acquired software that were created using programming languages, services, tools, and libraries that are supported by the cloud provider. The user does not have the ability to change or manage the cloud infrastructure, but does have control over the deployed applications. The user may also have the ability to change configuration settings for the application-hosting environment. The user may not modify the following cloud infrastructure features: network, operating systems, servers, or storage. (NIST, 2011)

Infrastructure as a Service (IaaS)

Infrastructure as a Service allows the user to provision the processing, networks, storage, and other fundamental computing resources. This model allows the user to deploy and run arbitrary software, which may include operating systems and other applications. The user does not have access to change or modify the underlying cloud infrastructure, but has control over deployed applications, operating systems, and storage. The user also may have limited control of select networking components such as host firewalls. (NIST, 2011)

2.1.3 Deployment Methods

The deployment methods of cloud computing can be broken into four distinct types of infrastructures. The different methods list who can and should be using the specified cloud infrastructure. These methods are: private, community, public, and hybrid. (NIST, 2011)

Private Cloud

The private cloud method of deployment has the infrastructure provisioned for exclusive use by one organization, such as a company with multiple business units (e.g., users). This method allows for the cloud to be owned and operated by the user organization, a third party, or a combination of the two. The cloud can exist at an on or off-site location. (NIST, 2011)

Community Cloud

The community cloud method of deployment has the infrastructure provisioned for exclusive use by a specific subset of users from organizations that have shared concerns, such as security requirements and compliance considerations. The community cloud may be owned and managed by one or more of the organizations in the community, a third party, or any combination of them. This method may exist at an on or off-site location. (NIST, 2011)

Public Cloud

The public cloud method of deployment has the infrastructure provisioned for open use by the general public. This method may be owned and operated by a business, government, or academic organization, or any combination of those organizations. This cloud method exists on the property of the cloud provider. (NIST, 2011)

Hybrid Cloud

The hybrid cloud method of deployment is a combination of two or more of the distinct cloud infrastructures listed above. This method allows for the different infrastructures being combined to remain unique entities, but they are bound by a standardized or proprietary technology. The technology enables data and application portability. (NIST, 2011)

2.2 Skill Sets for Cloud Developers

This section explores various skills and methods that cloud developers should be familiar with. For specific technologies relevant to cloud computing, see section 2.3.

REST

REST (REpresentational State Transfer) is an architectural style for Application Program Interfaces (APIs) which describes six constraints: uniform interface, stateless, cacheable, client-server, layered system, and code on demand ("REST API Tutorial" 2017). Most cloud platforms provide their own APIs that adhere to the REST guidelines, including AWS, Azure, and Google Cloud. For further discussion of these platforms, see Section **2.3.1 Frameworks/Platforms**. Since this style of API is prevalent in the top cloud platforms, it follows that cloud developers should be familiar with using REST APIs. However, there is currently a lack of standardization between REST APIs on the cloud, as platforms prefer to use their own implementations. There are currently attempts in the industry to make these APIs more standardized, to increase understandability and reusability across all cloud platforms. (Petrillo *et al,* 2016)

DevOps

DevOps refers to the use of various tools, methods, and architectures in software development to allow for greater automation, integration, and collaboration between software and operations engineers (Waxer, 2015). Thus, DevOps is more a cultural movement within software development rather than a reference to any specific tools. DevOps is an attempt to combine development and operations departments into one team, removing scenarios where one team would be forced to wait for the other team to fix a bug or implement a feature to continue their own work (Ellis, 2014).

DevOps can improve several aspects of software development. Since teams are not reliant on one another to continue their own work, overall development speed is increased. Companies report higher quality applications after implementing DevOps. Finally, DevOps directly increases the amount of collaboration within a company. (Bednarz, 2013)

DevOps itself is an umbrella term that includes several different types of tools. These categories include containers, configuration management, and continuous integration/continuous deployment, which are discussed further in Sections **2.3.4 Continuous Integration/Continuous Deployment**, **2.3.5 Containers**, and **2.3.6 Code Management & Monitoring**.

Microservices

Microservices are the key to the new complexity that businesses are experiencing due to the change in competitive software development (Felipe, 2016). Traditionally, programs were built in one code base, with the goal to create a single monolithic application that was executed. Software engineers used different programming languages and various modularity techniques to compartmentalize the source code into classes, modules or units to make it easier to maintain and debug, but the final resulting executable was expected to be a singular entity. Today, most

businesses want to implement changes to processes and customer experiences almost instantly. Microservices again fit this need. (Felipe, 2016)

A microservice is a component of a larger process that is well encapsulated on its own. The service can function on its own, without external dependencies, handling its own data storage and transactions. In the world of microservices, if one component were to fail or go down temporarily, the whole process would not be jeopardized. In the traditional model, a bug in one section of the program would affect the whole code base. (Richardson, 2017)

With the implementation of microservices, larger coding efforts may be delivered in a piecewise manner to an external source for larger testing efforts. Bugs that are found in the smaller components would be easier to spot and fix before doing end-to-end testing. Decoupling services allows larger, month-long, code development to be completed in shorter periods because multiple developers can work on several microservices in parallel. (Richardson, 2017)

Microservices can decrease the development time needed to deliver important business requested improvements and even develop new features from scratch quickly. There are development tools such as Spring Boot and Maven that work together to increase the efficiency of creating microservices. (Richardson, 2017)

Languages

Cloud developers should learn at least one of the object-oriented languages mentioned below in section 2.3.2. Developers should learn at least one scripting language, such as Node.js, which is supported by the top cloud platform providers, as discussed in Section **2.3.1 Frameworks/Platforms**. Ruby is supported and used to build many of the DevOps tools as listed in Section **2.3.8 Languages**.

2.3 Cloud Technologies

With the rise of Cloud Computing, the technologies that developers and companies must use have changed. The location of code is moving from local repositories to cloud platform hosted remote repositories. Large companies are charging for the services that work with developers to ease building, testing, and deploying code instead of developers manually doing all three in longer, more tedious steps. Programs are being decoupled into microservices that are able to be built and deployed faster.

In this section, we present research on the most prevalent technologies in cloud development. These technologies are the non-exhaustive list from our research that attempts to capture the state-of-the-practice in cloud development as of October, 2017. The results from our job listing research determined which technologies made the non-exhaustive list and can be viewed in Section **4. Results**. These include platforms, languages, databases, containers, DevOps tools, microservices, and version control software.

2.3.1 Frameworks/Platforms

Amazon Web Services

Amazon Web Services (AWS) launched in 2006 by Amazon.com as a secure cloud services platform. AWS offers computing power, database storage, content delivery, and other functionality that helps businesses scale and grow. As of August 2017, AWS includes multiple types of services, ranging from Computing to Code Management. Each service has several programs to choose from depending on the user's needs.

For computing and running code on the AWS Cloud, AWS Lambda is the option that best suits most developer's needs. As a serverless computing function-as-a-service platform, Lambda allows code to be run for almost any type of application or backend service without having to manage, ration, or administer servers. AWS Lambda allows developers to write applications that respond to changes in data or environment. (Close-up Media, Inc., 2014)

AWS Lambda has an auto scale feature that scales to the size of the workload based on each trigger of the code. Lambda charges for every 100ms that code is being executed and the number of times the code is triggered. When code is not running, there is no charge to the developer. (Amazon, 2017)

Lambda can be written in "Lambda functions" or in several third party languages. A Lambda function is always ready to run as soon as it is triggered, and includes configuration information along with the written code. The functions are "stateless" meaning that they have no underlying infrastructure that way multiple copies of the same function can be running in parallel as needed based on workflow. Lambda can run external libraries and currently supports Java, Node.js, C#, and Python code, with support for other languages coming in the future. (Amazon, 2017)

Azure

Azure launched in 2010 by Microsoft under the original name of Windows Azure. Azure is a cloud computing service created for building, testing, deploying, and managing applications and services. Offered as all three service models mentioned in Section **2.1.2 Service Models**, Azure supports multiple programming languages. There are currently 75 different products offered on Azure, ranging from several different Database types to virtual machines. (Microsoft, 2017)

For deploying infrastructure, Azure offers two virtual machine options for developers. The developers have the option to deploy using Linux and Windows virtual machines. Developers have the option to use two command-line tools, Azure CLI and Azure PowerShell. Azure offers containers with Kubernetes, microservice options and orchestration of containers with Service Fabric. Azure also offers serverless functions in a similar fashion to AWS Lambda. (Microsoft, 2017)

Azure offers several database options, in both NoSQL and Relational databases. Cosmos DB is the option for NoSQL databases offered. The relational databases are offered as a service

and those are SQL database, PostgreSQL database, and MySQL database. Azure includes Computer Vision API and Face API as well for AI and Cognitive Services. There are five third party languages supported currently by Azure. Those are Python, .NET, Java, PHP, and Node.js. (Microsoft, 2017)

Google Cloud Platform

Google Inc. launched Google Cloud Platform in 2011. The platform is a suite of services that run on the same infrastructure as the end-user products that Google offers, such as Google Search and YouTube. Google Cloud offers developers all three options of service models listed in Section **2.1.2 Service Models**, along with a few others not part of the NIST formal definition of Cloud Computing. Database as a service is provided with the Cloud Datastore service that provides a document-oriented database to users.

As of 2017, Google Cloud started to offer Functions as a service. This is similar to both Azure and AWS Lambda, which both provide serverless functions that can be triggered by cloud events. The platform has an Internet of Things service that allows for devices to be connected to the global device network. Currently Google Cloud supports the following languages: Go, Java, .NET, Node.js, PHP, Python, and Ruby. (Google Inc., 2017)

Google Cloud has its own cloud deployment manager, which allows one to specify all resources needed for the given application using a yaml document (Google Inc., 2017). This manager can also use Python or Jinja2 templates to configure common deployment paradigms that can be reused by multiple deployments. There is also an option to host all code bases on a private Git repository on the Cloud platform using Stackdriver. Developers can also connect GitHub or Bitbucket repositories to the internal Google Cloud Source Repositories. (Google Inc., 2017)

Hadoop

Launched in 2011 by Apache Software, Hadoop is an open-source software framework that is used for distributed storage and the processing of big data datasets. Hadoop is a cluster of computers built with the ideology that hardware fails and should be handled by the framework. While being written in Java, Hadoop is not a software for building code. Currently several cloud platforms are using Hadoop's software. The larger contributors include Azure, Amazon Web Services, Google Cloud, IBM, and Oracle. (IBM, 2017)

One of Hadoops's main functionality is data storage, which is done in the Hadoop Distributed File System. The other main functionality is stored data processing, which follows a MapReduce programming model. MapReduce is a job tracker that helps to schedule jobs to be run. Hadoop takes in large files, breaks them into blocks, and then distributes them across nodes in one of the clusters. In this model, the datasets are processed faster due to taking advantage of data locality. (Apache, 2017)

2.3.2 Databases

NoSQL

NoSQL, short for "not only SQL", is a database system that foregoes storing relations between data in tables in favor of storing large amounts of data through other means. This storage technique allows large data sets to be stored cheaply across servers, improving scalability and processing power. NoSQL is a solution to the lack of scalability inherent with traditional relational databases. As the industry shifts towards cloud development, this scalability is highly desirable. Types of NoSQL databases include Key-Value Store, Document Store, Column Store, and Graph Store. (Open Source FOR You, 2013)

NoSQL databases may support more than one type of storage model. For example, Amazon's DynamoDB supports both Key-Value Store and Document Store.

Key-Value Store

In a Key-Value Store database, keys are used to access groups of similar data commonly referred to as "blobs". It is the application's responsibility to understand exactly what is stored (Vishwakarma, 2017). Key-Value Store databases are simple to implement, have great performance, and has good scalability. However, it is impossible to query these kinds of databases based on stored values (McCreary, 2014). Example uses of Key-Value Store include user session data, user preferences, and shopping cart data. Key-Value Store databases include DynamoDB, Redis, MemcacheDB, and Riak (Vishwakarma, 2017).

Document Store

Document Store databases are similar to Key-Value Store databases, but have structured or semi-structured data instead of blobs (Vishwakarma, 2017). Document Store databases have great performance for searches. However, Document Store is difficult to implement, because if the data is not stored properly, performance and scalability can be poor (McCreary, 2014). Example uses of Document Store include e-commerce platforms, analytics systems, and blogging platforms. Document Store databases include DynamoDB, MongoDB, CouchDB, ElasticSearch (Vishwakarma, 2017).

Column Store

Column Store databases store data as columns rather than rows. Columns are grouped into families with similar data that is often accessed at the same time (Vishwakarma, 2017). Column Store databases have great performance on simple searches and great horizontal scalability. Downsides of Column Store are poor performance for complex queries and, like Document Store, a poorly designed implementation will diminish any benefits (McCreary, 2014). Example uses of Column Store include content management platforms, services with

counters, and services with many write requests. Column Store databases include Cassandra and Hadoop Hbase (Vishwakarma, 2017).

Graph Store

Graph Store databases store data in nodes along with relationships between different nodes. While traditional databases describe each possible relationship in foreign keys, Graph Store can describe any relationship between two nodes spontaneously. In this type of database, queries are graph traversals (Vishwakarma, 2017). Graph Store databases are ideal for representing relationship networks and searching through these networks. Graph Store should be avoided if the data is not sufficiently related and can have issues with scalability (McCreary, 2014). Example uses of Graph Store databases are fraud detection and social networks. Graph Store databases include Neo4j, ArangoDB, OrientDB (Vishwakarma, 2017).

Relational Databases and NewSQL

Traditional relational databases are still desirable because they subscribe to ACID, properties which ensure the consistency of data within a database, while NoSQL databases do not. ACID stands for Atomic, Consistent, Isolation and Durable, the key properties that ensured the integrity of databases in the face of concurrent access. One of the largest weaknesses of relational databases is poor scalability with large data sets. In cloud computing environments where data is large, relational databases struggle to compete with NoSQL databases. One of the leading relational databases is PostgreSQL. (Hammes *et al*, 2014)

NewSQL is a recent movement to attempt to implement the scalability of NoSQL while maintaining ACID. Theoretically, NewSQL databases would provide excellent performance and reliability for data of any size. Currently NewSQL databases have failed to gain a significant market share, as large companies prefer to develop and improve their own databases. However, NewSQL may be a contender in the future. (Pavlo & Aslett, 2016)

2.3.3 Configuration Management

Configuration management tools allow users to configure systems remotely either through a node system managed by a master server or through pushing updates to other systems via SSH. Through configuration management, it is possible to quickly ensure that all systems on a network are up to date regarding a variety of software. (Heller, 2017)

Puppet

Puppet is a configuration management tool introduced by PuppetLabs in 2005. Puppet uses a master-client architecture, in which a master server controls several nodes on a network. Using a ruby-based declarative language, the developer can write scripts to define how the system should be configured. Then, the desired state is deployed to the nodes on the network automatically and any differences in the state of a node and the desired state are reported back to the master. (Miglierina, 2014)

Puppet has multiple characteristics that make it highly desirable. Puppet is in and of itself a comprehensive tool for almost every system. Though Puppet is complex due to its comprehensiveness, its custom declarative language makes it easier to learn and use than other similar tools. As one of the oldest competitors in the market, Puppet has a well-developed GUI compared to other tools. Additionally, Puppet has a large active community which makes it much easier to find specific user-made scripts. (Torberntsson & Rydin, 2014)

Some drawbacks of Puppet include its difficult installation and performance. The initial configuration of Puppet on a system can be more difficult than other tools because of its complexity. Additionally, while its custom language is easier to learn than traditional Ruby, it may be more difficult to learn than a Python-based tool. Finally, Puppet has worse performance than other, more lightweight tools. (Torberntsson & Rydin, 2014)

Chef

Chef is a configuration management tool introduced in 2008. Chef's master-slave architecture is similar to that of Puppet in terms of deployment. Scripts are uploaded to the master node or server, and then deployed to the rest of the network's nodes from there. Chef utilizes "cookbooks" which are Ruby scripts for configuration and deployment. The goal of these cookbooks is to allow users to use scripts created and field tested by other users, reducing development time. (Benson *et al*, 2016)

Chef's strengths largely lie with its technical ability. Chef has the best performance of the configuration tools discussed in this paper (Benson *et al*, 2016). Chef is the most flexible of these tools as well, giving developers greater control over most aspects of configuration and deployment. The use of scripts rather than a declarative language allows users to share tested "cookbooks" easily. (Torberntsson & Rydin, 2014)

Chef's largest weakness is that it is difficult to use. Since it only uses Ruby, Chef will be difficult for users without intimate knowledge of Ruby. Furthermore, since Chef uses scripts, users must be careful about ordering in their deployments. A declarative language, such as the one used by Puppet, will determine the correct deployment order for the user (Torberntsson & Rydin, 2014). Lastly, though Chef implements a community-based approach, its active community is much smaller than that of Puppet (Benson *et al*, 2016).

Ansible

Ansible is a configuration management tool introduced in 2012. Ansible uses an agentless architecture, meaning it does not require an agent to be installed on nodes. In other architectures, agents are used by the master node or server to control the slave nodes. Instead, Ansible simply executes its code over SSH. (Benson *et al*, 2016)

Ansible's unique architecture results in several unique strengths. Ansible is much simpler to install because agents do not need to be installed on each node. Additionally, the lack of nodes can reduce network traffic, as agents need to report back to the master node. Ansible offers

greater security options because it uses SSH. Ansible also supports modules in several languages (Benson *et al*, 2016).

On the other hand, Ansible experiences issues with its performance and UI. Ansible has the worst performance of the tools discussed in this paper, especially on larger systems (Benson *et al*, 2016). Ansible is also not as simple to set up on larger systems. Both of these factors severely inhibit Ansible's scalability. Ansible also has a poorly developed UI. The UI is not tied directly to the command line interface, which can cause delays and usability issues. (Torberntsson & Rydin, 2014)

SaltStack

SaltStack is a configuration management tool introduced in 2011. It has a similar master-client architecture like that of Puppet and Chef. However, SaltStack provides support for multiple master nodes, including masters of other master nodes. This capability creates a tiered hierarchy of nodes. (Benson *et al*, 2016)

SaltStack's tiered hierarchy gives it many competitive strengths. Tiers allow users to configure specific groups of nodes, which results in excellent scalability. Compared to Chef and Ansible, SaltStack required less code to achieve the same result, meaning it is much easier to program for. Additionally, SaltStack experienced much better performance than Ansible and only moderately worse performance than Chef. (Benson *et al*, 2016)

However, SaltStack has some management level issues. SaltStack lacks proper monitoring tools for vast systems of tiered nodes (Benson *et al*, 2016). SaltStack also lacks a robust GUI. Though it is more lightweight, SaltStack does not have as comprehensive tools and solutions as more complex software, such as Puppet (Torberntsson & Rydin, 2014).

2.3.4 Continuous Integration/Continuous Deployment

Continuous Integration/Continuous Deployment tools allow for continuous integration of code by many developers from different machines. Continuous integration involves constantly and automatically pushing updates, provided they pass given test cases. When paired with configuration management, it is possible to keep an entire network continuously updated automatically. (Heller, 2017)

Jenkins

Jenkins is a continuous integration tool introduced in 2005. It is widely considered the most popular automation server in use, with 511,446 active Jenkins nodes as of February, 2017 (Database and Network Journal, 2017). Jenkins works with version control tools, such as git, to create an automated version control system. Developers can commit builds as in normal version control. Builds can also be scheduled to automatically commit at certain times.

By automating this system, developers can spend less time managing their code base. Anecdotal evidence suggests that Jenkins is difficult to manually configure, so automation is recommended. Jenkins also provides developers with dashboards and other utilities with which to properly monitor their version control system and modify it accordingly. (Manufacturing Close-Up, 2016)

2.3.5 Containers

A container uses virtualization to create an environment in a sort of package that is easily transferrable between different platforms (Crosman, 2015). Containers are similar to virtual machines in that they simulate an operating system for programs to be run on. However, containers have the additional benefit of being far more lightweight and less resource intensive than traditional virtual machines (Silver, 2017). Containers are useful for cloud development, and some cloud platforms, such as AWS and Azure, have native container support (Kozhayev & Sinnott, 2017).

Containers circumvent the configuration required to get traditional software to work on multiple environments. Each environment has many different variables that can interfere with software running in the exact same way. Containers eliminate these extra variables, causing a direct increase in reproducibility. (Silver, 2017)

Containers also improve development speed, particularly with cloud developers. Developers are not required to consider what platform they are developing an application for. Rather, they can develop the application for just the container and be confident that it will run on every system, provided the container supports those systems. One use for this ability include transferring applications between different clouds as required. A second use is that applications can be uploaded to a private, internal cloud and be developed on by any machine. This capability is particularly useful for developers working for organizations in which the cloud is kept private for security, including banks and the government. (Crosman, 2015).

Containers complement DevOps well in that they provide tools for both software development and operations work. Software developers can program and test on environments the same as the live server, reducing configuration issues in the future. Meanwhile, for those in operations, containers reduce the need to support several different systems and allow for testing changes in the same environment that developers will be using. (Ratan, 2017)

However, containers currently lack security features. Security must be decided around containers, not within one. Some vendors offer security add-ons to complement their containers. As it currently stands, security is not a reason to choose containers, but it may be in the future. (Crosman, 2015)

Docker

Docker is a container tool developed by Docker, Inc. Docker consists of three versions: developer, operations, and enterprise. Docker creates an image that contains both the application

to be contained, the developer's environment, and a configuration file created by the developer. The image is then turned into an executable package which can be exported to other environments. From there, the executable package can simply be run on other environments. (Docker Inc., 2017)

Docker is currently the leading container tool in the industry. A survey of more than one thousand IT professionals revealed that 35% of respondents were currently using Docker, and 32% had plans to implement Docker. These numbers far outmatched that of any other DevOps tool. (Yegulalp, 2017)

Docker experiences several advantages that make it a top competitor among container tools. Images produced by Docker tend to be small and produce virtually no additional overhead (Kozhayev & Sinnott, 2017). Docker requires minimal application runtime, which allows for faster deployment. Docker contains built in version control and supports remote repositories. Docker is supported among a wide variety of environments and can be easily integrated with other infrastructure tools, such as AWS, SaltStack, and Jenkins. (Ratan, 2017)

The largest weakness of Docker is its complexity. Since it attempts to integrate so many features, learning Docker can be a daunting task. Furthermore, configuring the Docker environment can be difficult for the inexperienced. If configured incorrectly, Docker may not work the same as the desired environment. (Ratan, 2017)

Kubernetes

Kubernetes is a container orchestration tool developed by Google. Kubernetes supports image-based application containers, and even offers integrated support for Docker. However, Kubernetes also acts as a management tool for automating, deploying, and scaling containers. Kubernetes groups application containers into logical groups, called pods, based on application and resource requirements for simpler management. Each pod is assigned its own IP address that can be reached by any other pod. Additionally, containers in the same pod are generally collocated on the same physical machine. (The Kubernetes Authors, 2017)

Kubernetes' management tools produce a number of enhancements to application containers. By grouping containers into pods, Kubernetes makes the system easier to manage. Additionally, these pods improve scalability by creating groups that can be managed similarly. Furthermore, pods reduce resource consumption by grouping applications based on hardware needs. (Pahl, 2015)

Despite the benefits, pods have some drawbacks as well. The system loses overall flexibility when applications need to be collocated on the same physical machine with their data. Kubernetes systems also require advanced network support to run efficiently, since each pod is hosted at its own IP address. (Pahl, 2015)

2.3.6 Code Management & Monitoring

Git

Git is an open sourced, free code management tool that was created in 2005 after BitKeeper revoked the tool's free to use status. The community that created the Linux kernel was the largest user of BitKeeper's tool. This community reverse engineered the functionality they learned from BitKeeper in order to build their own system. The goals of the new system were speed, being fully distributed, simplicity, ability to handle larger projects (like the Linux kernel) efficiently, and a strong support for non-linear development (having thousands of parallel branches). (Chacon *et al*, 2014)

Git as a tool allows for code to be used by multiple developers locally and remotely using its branching and merging features. The branching model allows for frictionless context switching to different code bases or versions of the current code base. With the branching off of the main branch to create different features that may or may not go into production, role-based codelines can be kept so that a branch can be used solely for what is going into production. This minimizes broken code from accidentally being deployed into the working production environment. The branching also allows for experimentation because local branches based on the remote branch can be created and deleted without harming the remote branch. (Chacon *et al*, 2014)

Git allows for multiple local branches to be created by the same or different developers from the same shared remote repository. These remote repositories allow for backups to be created and for different types of workflows to be implemented. With each developer having the ability to create and push local branches, Git captures the time and developer who makes any type of event, be that a commit, a push, or a merge. This allows for backtracking if a feature goes missing, or needs to be removed from a master branch that is going to go live into production. (Chacon *et al*, 2014)

Git has a staging area that gives developers a space to check over their commit files and add an optional commit message before completing the commit. This area also allows for the staging of selected files of a project, and only files that have been modified. (Chacon *et al*, 2014)

Artifactory

Artifactory is a version control tool for binary artifacts, such as Jar and War files. Files may also be stored in Artifactory as a way of allowing other project teams to use the files across companies with minimal cross-team interactions. Artifactory allows for the storage of files that have already been tested so that other teams can use without having to incorporate testing efforts for that service. Jenkins can be connected so that when a clean build is made, the metadata for the builds are automatically integrated into the file being built. (Ford, 2014)

Artifactory supports projects from many different tools, such as DevOps tools (Puppet, Chef, etc.) and various programming languages (Ruby, Java, Python, etc.). The tool also

connects to remote repositories so that developers and DevOps teams do not have to search and download copies of configuration services to their local machine. (JFrog, 2017).

2.3.7 Microservices

Spring Boot

Spring Boot is a framework added to the Java programming language. The framework can currently be installed into the Eclipse IDE for Java, or can be downloaded in an IDE that mirrors Eclipse called Spring Tool Suite. The framework allows for easily importing other projects, and includes libraries for connecting to AWS and the Netflix cloud platform (Hikari, Eureka, etc.) for cloud hosting. Spring Boot has APIs for most development needs, so minimal coding is needed to create new microservices, thus minimizing time to create and debug lengthy code bases. (Pivotal Software, 2017)

Spring Cloud

Spring Cloud is a project that was created using Spring Boot that helps connect projects to a cloud platform. The project has several branches that developers have created for Pivotal Software. The different branches help to connect to different cloud platforms. Some supported platforms include Netflix OOS, Cloud Foundry, and AWS.

Other components to the project include different frameworks built for the many facets of the developing community. Spring Cloud includes libraries and APIs for building projects for business contracts, projects that connect to external servers, and many more. (Pivotal Software, 2017).

Maven

Maven is an Apache project that helps with the compiling of code. It is a project management software and comprehension tool. The software is based on the project object model (POM). Maven manages a project's build, reporting, and documentation from one central piece of information, known as the POM file, using XML.

Maven includes a framework for best practices for setting up projects to make them easier to start. The software also works with multiple projects at once, allowing multiple microservices to be run in parallel without issue. Maven allows for plug-ins to be implemented in one location for the all files of a project to use. This feature decreases bugs due to forgetting to implement an API in one file. (Apache Software Foundation, 2017)

Gradle

Gradle is an open-sourced project that helps with the building of code. It is a project management software and comprehension tool. The software is built upon both the Apache Ant and Apache Maven projects. The documentation for build configurations for Gradle use a

Groovy-based domain-specific language. Gradle has an internal set of APIs and plugins that allow developers to implement code and build quickly. Gradle has the capacity to be used by mono and multi repository projects, and was built to scale for large projects. (Gradle Inc., 2017)

2.3.8 Languages

Python

First released in 1991, Python has since had two updates to modernize the language to keep up with new technologies. Python is an interpreted language, making it easy to read, understand, and write (Python Software Foundation, 2017). According to the Python Software Foundation, as of 2017, Python is used by upwards of tens of thousands of users. Python is a universal programming language that is currently being supported by the top three cloud provider platforms mentioned in Section **2.3.1 Frameworks/Platforms**.

Java

Having been around since 1995, Java is one of the most used programming languages in cloud computing. The top three cloud platform providers, mentioned in Section **2.3.1 Frameworks/Platforms**, support Java for building programs and deploying using their services. Java 8 was upgraded to fit the cloud, allowing programs to fit wide-scale cloud deployment. The upgrades that support cloud development were multitenancy, the ability to run multiple applications on the JVM safely, and modularity, allowing for the JDK to be reorganized to be cleanly defined into interdependent modules. (SyndiGate Media Inc., 2011)

Java will have some drawbacks and advantages depending on which service model, see Section **2.1.2 Service Models**, the developers will be deploying in. Using Java for IaaS, the cloud can make deployment and testing easier for developers due to the built –in services that the cloud providers may have. For PaaS, the developer does not need to worry about physical or virtual machines, the application server platform, and the database systems. The platform provider is now supplying and in control of these resources. Using PaaS, the developer does have to plan the deployment platform out because there is no standard similar to Java EE currently. Development has to be done using the API/SDK provided by the platform vendor. (Panda, 2011)

For SaaS, developers no longer need to handle code management or continuous integration services in-house, because the provider handles those services. For all three service models, the writing of code remains the same. Deployment and testing change depending on service model. In the cases of IaaS and SaaS, deployment and testing are easier, but in PaaS, a small amount of accommodation is required. (Panda, 2011)

Ruby

Released in 1995, Ruby is a scripting language that is also a powerful object-oriented programming language ("About Ruby", 2017). Only one of the top three cloud platform

providers, Google Cloud, currently supports Ruby. Developers building cloud applications cannot program in Ruby easily due to the lack of support by the providers. However, many cloud DevOps tools are written and maintained mostly using Ruby. These DevOps tools are discussed further in Section **2.3.3 Configuration Management**.

Node.js

Node.js was created in 2009 and is a version of JavaScript that is used primarily for creating web servers. Node.js is an event-driven language that works well with the event-driven triggers used in serverless cloud computing platforms (Node.js Foundation, 2017). All three of the top cloud platform providers mentioned in Section **2.3.1 Frameworks/Platforms**have Node.js as the only scripting language being supported.

Go

Go is a programming language that had its first stable release in August of 2017, but first appeared in 2009. The language is a compiled, statically typed language modeled after C. The language includes limited structural typing, garbage collection, and has features for both memory safety and concurrent programming. (Golang Blog, 2017)

Go was written expressly for the cloud by incorporating useful features of several popular programming languages. Go developers also considered the failures of the different languages, especially in regards to cloud computing. The most improved upon feature is the ability to program concurrent processes more easily than before. According to Rob Pike, one of Go's designers, Go will be easier to read, write, maintain, and understand. (Asay, 2014)

Bootstrap

Bootstrap is a front-end development framework that can be used with HTML, CSS, and JavaScript. Bootstrap was created in 2011 as a project by Matt Otto and Jacob Thornton as a way of creating web and mobile applications that looked professional and were consistent across the industry. Bootstrap provides various API's that help developers improve the visual elements of a webpage. The fourth version of Bootstrap released in 2017, with new features added to the framework. The code is open source and the code is licensed under the MIT license. (Github Inc., 2017).

2.4 Industry Initiatives

In the past several years, companies have been embracing the cloud computing world. The companies vary greatly across all industries, and seem to vary in what they use the cloud for as well. While there are some big name companies backing cloud computing, some industries are not satisfied with their service.

Ford Motor Company jumped in to using Cloud Computing in their hybrid cars in 2011. For uses cloud technology enhance efficiency in three main areas of vehicle performance: intelligent operation, intelligent routing, and intelligent driving. The technology is used to detect driver behaviors to optimize the control systems of the vehicle and improve fuel or hybrid-electric efficiency. In the hybrid version of the Ford Escape, cloud technology is used to detect lower emissions zones and switch the car over to all-electric power. (Spiegel, 2011)

In early 2015, Walmart moved all of their e-commerce operation on the cloud using OpenStack. This move included more than 100,000 cores and several petabytes of storage. Walmart's senior director of cloud operations and engineering, Amandeep Singh Juneja, stated that Walmart was due for a technology upgrade to their e-commerce platform. The previous platform was a legacy Art Technology Group (ATG) solution on top of a scale-up production infrastructure. Juneja supports OpenStack because the technology is mature enough to use at such a large scale in production. (Knorr, 2015)

The online television streaming service Hulu has decided to move to the cloud. The company decided to use Amazon Web Services as their cloud provider. The main reasons Hulu chose AWS are that they wanted an agile, scalable, and cost effective infrastructure to support the addition of at least 50 more channels that users can stream live starting in May 2017. (Entertainment Close-up, 2017)

As of May 2016, the United States House of Representatives has been banned from using any Google Cloud applications hosted on appspot.com due to an increase in phishing scams. Early May 2016, two individuals had fallen victim to phishing scams that were sent via word document email attachments. Reuters reported on the ban a week after the incidents occurred, and included the ban on Yahoo mail that occurred in 2015. The only reason, reported by Reuters, is that the FBI warned about an increase in viruses from these cloud platforms. There was no report on if other cloud platforms were also warned against. (Volz, 2016)

2.5 Cyber Security and the Cloud

Cloud computing brings about a new threat to security of not only data, but also devices, and money. When looking into cloud computing, the type of data users will be working with and possibly storing can impact the cloud provider they use and if they even move to the cloud. Although security is important, it was not the focus of this project. Thus, security is only covered briefly.

In 2013, KLAS research group surveyed several healthcare companies to see how they rated using cloud technology in their workplace. Users of the cloud rated on average 4.5 out of 5 on security using off-site IT services. On the other hand, 66% of non-cloud users were wary of security, ranking it their greatest concern with cloud technology. (Health and Management Technology, 2013)

The company Microchip, in 2016, created an end-to-end solution that will allow Internet of Things (IoT) devices to easily set up connections to Amazon Web Services' cloud platform. The process of connecting to AWS using an IoT device is long and can be complicated for some manufacturers. The multi-step process to set up devices with the security encryption is needed to comply with the advanced security model. The process is simplified by soldering Microchip's service onto the IoT device. (Electronics For You, 2016)

Before Microchip's product, or without, the manufacturer must pre-register their security authority with AWS servers to establish a trust model. Then for each IoT they must generate a unique encryption key that are linked to the pre-registered security authority. The unique key must be kept a secret for the life of the device, or risk the device being compromised which can lead to all devices being compromised. Microchip's solution handles the whole process during production. The customer meets the security standard set by AWS authentication model and connect to the AWS IoT platform easily. Both of these happen during the evaluation and engineering phase of the device. During the prototyping phase, Microchip's device helps with meeting the security standards. Lastly, the devices are customized for production; this will ensure that customer information will be secure. (Electronics For You, 2016)

3. Methodology

The goal of our project is to create a robust training structure for full stack cloud developers. To achieve this goal, we executed the following process:

1. Evaluate Current Cloud Development Positions
2. Identify Initiatives by Leading Companies
3. Research Cloud Technology
4. Research Cyber Security
5. Build an Image Classification Service

3.1 Evaluate Current Cloud Development Positions

We will interview career development professionals to gain insight into best practices for clear and concise job searches. Then, we will incorporate their advice in order to select the best websites for our job search. We will then take industry keywords found from our background research to search for cloud computing positions in the Boston area. As a tech center, Boston will provide us with a large enough sample size while still allowing us to narrow the scope of our project. Using this sample size, we will collect the most common technical skills listed in the requirements section of these job postings. Jobs not directly related to cloud computing, such as marketing jobs, were excluded. Technologies not related to cloud computing, such as networking, were excluded.

Additionally, we will attend the WPI Fall Career Fair and speak with hiring managers there. From these interviews, we will be able to discern an expert opinion on the requirements of cloud development. These managers will provide us with additional detail not available with just job postings. We will ask each manager the same set of questions to provide us with consistent data. For the full list of questions please see Appendix A.

3.2 Identify Initiatives by Leading Companies

Our second objective is to research the projects performed by leading companies in cloud computing. We will search databases to find articles. We will also look at company newsletters, announcements, and blogs. From these projects, we will gain a better understanding of the direction the industry is trending towards. Additionally, this research will help us understand the type of work that students could expect once they graduate.

3.3 Research Cloud Technology

As a team, we will research the current and potential future of cloud technologies. Research into these areas will allow us to provide a clear understanding of what a full stack cloud developer will be doing in the workplace. The types of technologies we will research will

include, but not be limited to, microservices, databases, code management, languages, and frameworks. We will focus on how cloud computing causes potential change to currently established technologies.

Using database searches, our research materials will include Trade Journals, academic papers, news articles and reports from companies, and online forums for the different technologies. These methods of media and information will house the most up to date information for this rapidly growing facet of technology.

3.4 Research Cyber Security

Lastly, our research will involve how cloud computing changes how we as a society view cyber-security. We will search databases to find academic papers regarding cyber-security currently interacts with the cloud, and potential changes to the future of cyber-security in cloud computing.. These resources will also be paired with how the future of cyber-security is currently being adapted to account for a large influx of data and other personal information being stored the cloud.

3.5 Cloud Application Reference Model

In this section we present the reference model we used to implement the proof of concept project in Section **3.6 Image Classification Service**. Each tier contains a technology or method that we have determined to be necessary for building a cloud-based application. The tiers are in descending order of importance relative to cloud computing. In each tier, we present why it is necessary for cloud computing, any dependencies on other tiers, and some of the choices available that were previously discussed in this paper.

We discuss our choices further in Section **3.6.2 Implementation**. We discuss dependencies of specific technologies further in Section **5. Discussion** of this paper.

Platform: Platforms form the base of a cloud-based application and determine several choices for other tiers. Choices: AWS, Azure, Google Cloud

---

Database: Cloud-based applications often have intensive technical specifications since they often work with vast volumes of data. A database that fits these specifications is necessary for good performance. Certain platforms have native support for databases which makes them easier to implement on those platforms. Choices: Key-Value store, Document store, Column store, Graph store, relational

---

Configuration Management: It is too difficult to manage the large systems used in cloud computing without DevOps tools. At least one configuration management tool should be used. Choices include Puppet, Chef, Ansible, and SaltStack and are platform independent.

Continuous Integration/Continuous Deployment: Continuous integration tools can help large teams manage their code automatically. One option for this is Jenkins.

Container: Containers allow for easy transfer of applications between cloud environments. Most container tools, such as Docker and Kubernetes, are supported on the three platforms mentioned.
Code Management: Developers need a standardized method of code management to successfully maintain a codebase. Code management tools exist largely independent of other technologies. The methodology of code management is more important than any specific technology.
Microservices: Implementing a microservice model improves application development, though it is not cloud specific. The three cloud platforms include API gateways that facilitate the implementation of microservices. In order to implement microservices without using a platform's native API gateway, a third party software, such as Spring Cloud, is required.
Programming Language: The language(s) used depend on which languages are supported by the platform chosen. If implementing a microservice model of development, each module can be programmed in a different language, making the choice a personal preference for the developer.

Figure 3.1. Reference Model for Cloud Application.

3.6 Image Classification Service

We will be implementing cloud technologies to create an image classification service as a proof of concept project for our training manual. The end program will have a system for classification of images, be able to upload images for classification, and be able to search for already classified images. The application will have two types of admin accesses, one for the company and the other for moderators. The company will have the opportunity to monitor the user activity of the application and be able to have a reward system for users that classify images. The moderator will have the permission to block and allow images and ban users that violate the expected uses for the application.

3.6.1 Use Cases

We identified three types of personas for use cases for our application. They are end users, moderators, and the company.

End Users

E1. As an end user of the image classification system, I want to be able to search for images by keywords with results organized by relevance to the keywords, so that I may find images related to what I want in a short time frame.

E2. As an end user of the image classification system, I want to be able to upload images to the cloud to be able to access them later, so that I can view the images on any device.

E3. As an end user of the image classification system, I want to be able to classify images that have been uploaded by adding keywords, so that I can earn rewards given by the company.

E4. As an end user of the image classification system, I want to be able to have images be able to be classified as the same keyword more than once, so that the image will have a better chance of being in the search results for that keyword.

E5. As an end user of the image classification system, I want to be able to have a personal account to log into, so that I can track my uploaded images and points for classifying images all in one place.

Moderators

M1. As a moderator of the image classification system, I want to be able to block images that are not allowed, so that end users may have a clean and enjoyable experience.

M2. As a moderator of the image classification system, I want to be able to allow images that are not banned, so that end users may get their images classified.

M3. As a moderator of the image classification system, I want to be able to ban users that intentionally classify images incorrectly, so that end users using the application correctly
may get their images classified properly.

M4. As a moderator of the image classification system, I want to be able to ban users that intentionally classify images incorrectly, so that the search results do not return incorrect images.

M5. As a moderator of the image classification system, I want
to be able to ban end users that violate the acceptable images guidelines, so that end users using the application properly do not have to see unacceptable images in the classification system and search results.

The Company

C1. As the company that owns the image classification system, I want to be able to provide end users that classify images or upload images with a reward system, so that the end users will feel appreciated and continue to use the service.

C2. As the company that owns the image classification system, I want to be able to monitor the end users' use of the application, so that I can see if the service is worth the business investment.

C3. As the company that owns the image classification system, I want to be able to monitor the end users' use of the application, so that I can modify the cloud service to keep up with demands for the different aspects of the application when needed.

3.6.2 Implementation

In the following section, we discuss which cloud computing technologies we chose to implement to create the image classification system, based on the research explained in Section **2.3 Cloud Technologies**. These choices were made in accordance with our reference model in Section **3.5 Cloud Application Reference Model**. Each decision was made taking into account the current WPI teaching format for Computer Science Majors and the current trends in cloud computing technologies. We discuss the implications of selecting different technologies within each group in Section **5. Discussion** of this paper.

For this project, we used a website hosted on AWS' S3 service. We chose a website as the simplest way to present a front-facing UI that could be connected to the various services we intended to use. This decision is largely trivial and there are many alternatives that still follow our reference model.

Platform

We used AWS as the cloud platform provider. Most job applications we researched listed AWS as a skill they were looking for in the candidates. AWS also allows students to use their Educate tier, which supplies a free credit code to be able to use features that are not free. Additionally, AWS provides most of its services for free under certain storage and computation constraints through the free tier. AWS supports the most variety of programming languages, giving students more options to program in. AWS has a large selection of integrated utilities including user management, API creation, monitoring tools, and more. These utilities allow students to gain a wider range of experience than other platforms.

The other options that we decided to not use were Microsoft Azure and Google Cloud Platform. Most job listings preferred AWS over both of these options. Both platforms support Java and other languages, but are not as many as AWS. Both platforms also do not offer a student version for learning and academic purposes meaning that students would have to pay out of pocket from the start using these platforms.

Database

We decided to use NoSQL as it is more scalable than SQL. The database we used was DynamoDB because it has built in support for AWS and cloud computing, making it easier to connect our project to the database. DynamoDB supports both key-value store and document store models. Key-value store databases are vertically scalable and simplest to implement.

Configuration Management

For automation and configuration management, we decided to use Puppet. Puppet's robustness means it is better at management than any other single DevOps tool. Additionally, we decided that the Puppet command language would be easier to learn than some of the other tools. However, implementing Puppet is a stretch goal for our project, as it will not have the most meaningful impact on a two person team.

Continuous Integration/Continuous Deployment

While the most used tool for continuous integration has been Jenkins, this tool is most helpful for larger teams that have many developers committing code and making builts to deploy. Since we are a team of two students, this tool is not within the scope of our project and thus we will not be using a continuous integration tool.

Container

We chose to use Docker for the container tool. Docker is supported by AWS making it easier to incorporate into the project. Kubernetes is also supported by AWS, but with the small scale of our project we decided that the functionality of moving containers around different systems in a network is not within the scope of our project. We did not find any competitor container tools that we needed to choose between.

Code Management

We decided to use Git as our code management system. Most WPI classes mention GitHub and using Git for repositories, so this coincides with WPI teachings. We used BitBucket for our private repository hosting platform and Sourcetree as the UI. We made our own branches from our master branch and merged into that for end-to-end testing. We also had a release branch that held only code that had been through end-to-end testing and passed that would go into our production environment.

Microservices

We decided that using the microservices framework was best for our project so that we could build different components of the project and unit test without having to build the whole system. Microservices are also easy to implement using the AWS API Gateway.

Languages

We decided to use AWS as our platform, as noted above. Within this platform there are several services that require specific languages. AWS Lambda and the AWS API Gateway services have been easiest to use Python and JavaScript for the Lambda function creation. Additionally, we used JavaScript for the connections between the website and the API Gateway. CSS and HTML were used for the development of the website.

4. Results

4.1 Overview

We collected data on job listings using CareerShift, which pulls job postings from the internet based on keywords and filters. These sources include popular job boards as well as company websites. The below graphs show our data for the frequency of technologies and skills listed in our relational model in Section **3.5 Cloud Application Reference Model**. Our sample size of job listings was 67.
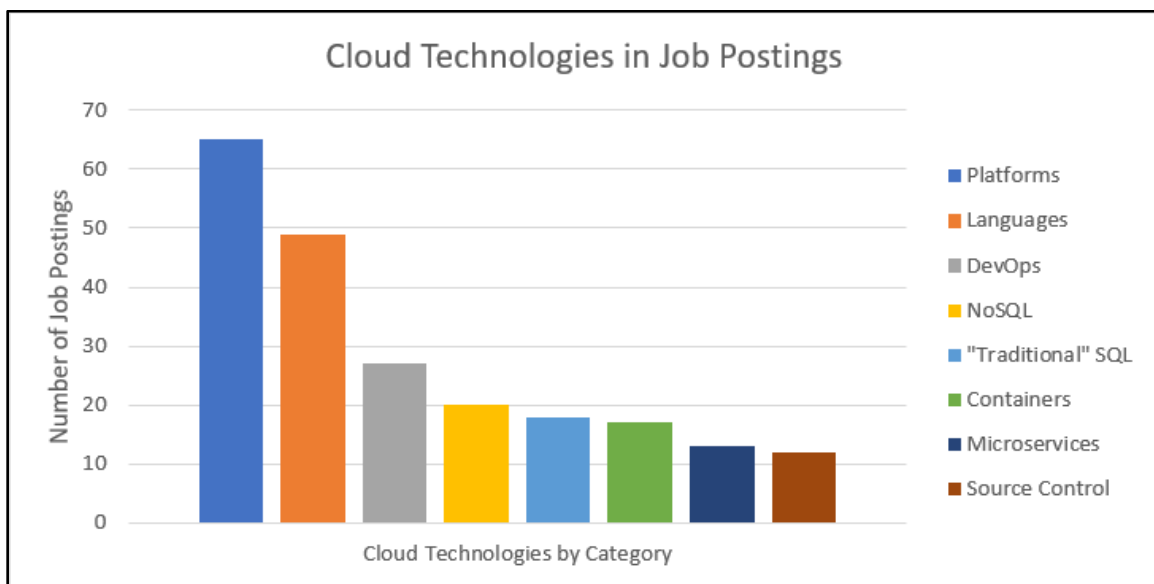

Figure 4.1. Bar graph of technologies listed in job postings.

Figure 4.1 shows the distribution of the technology categories previously mentioned in Section **2.3 Cloud Technologies**. 67 job postings (97% of total) requested Platform skills. 49 job postings (73% of total) requested programming language skills. 27 job postings (40% of total) requested skills using DevOps tools. 20 job postings (30% of total) requested NoSQL database skills. 18 job postings (27% of total) requested relational ("traditional") SQL skills. 17 job postings (25% of total) requested skills using containers. 13 job postings (19% of total) requested knowledge of microservice methodology. 12 job postings (18% of total) requested source control skills.
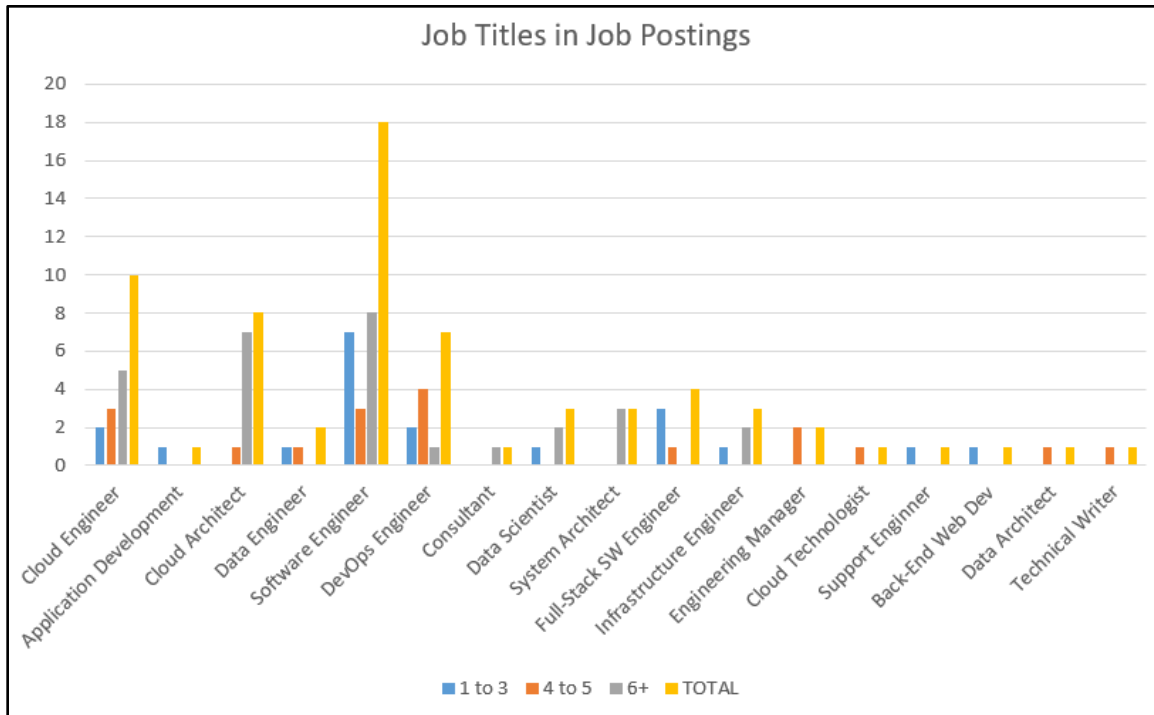
Figure 4.2. Job Titles in Job Postings

When searching for cloud computing jobs, we took note of what titles the companies were supplying for the keyword "cloud computing". Figure 4.2 displays the breakdown of position names seen then within those titles how many years of experience the job posting required. As seen above, the three most common job titles seen were Software Engineer, Cloud Engineer, and Cloud Architect, with the years of experience varying. Software Engineer was seen 18 times (26%), Cloud Engineer was seen 10 times (15%), and Cloud Architect was seen 8 times (12%).
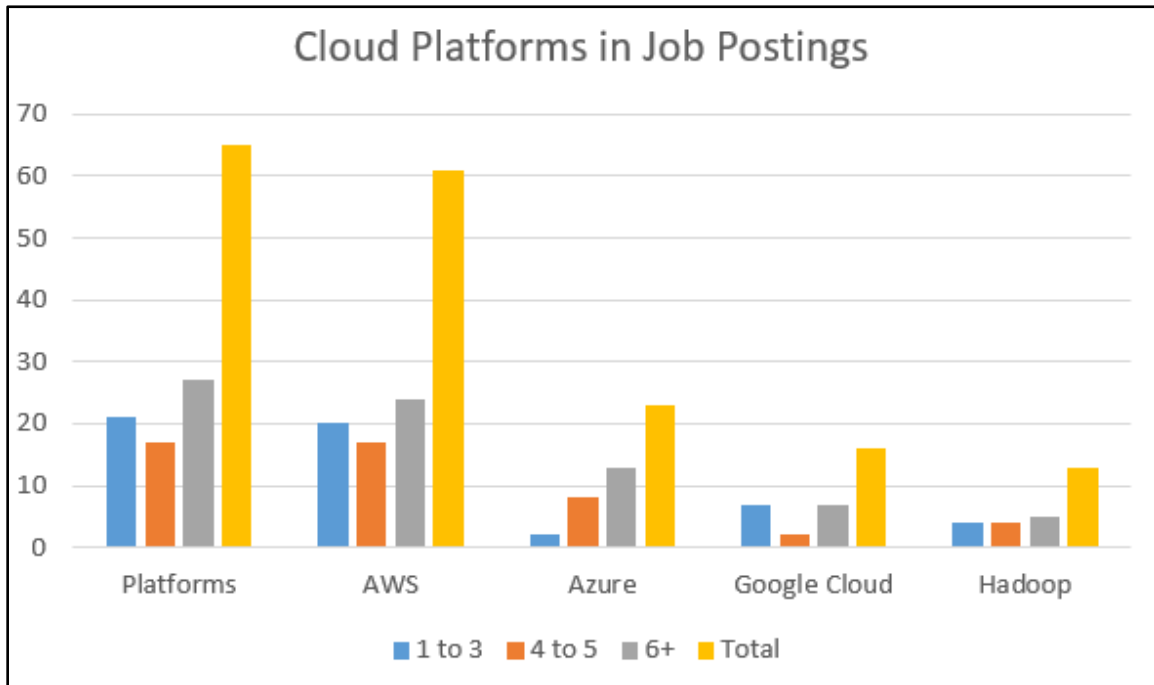
4.2 Platforms

Figure 4.3. Cloud Platforms in Job Postings

Figure 4.3 displays the frequency that the cloud platforms were requested by job postings. 65 job postings (97% of total) requested any platform type. Of these postings, the different platforms were requested differently. AWS was requested 61 times (94%), Azure was requested 23 times (35%), Google Cloud was requested 16 times (25%), and Hadoop was requested 13 times (20%).

4.3 Databases

For the results of the database research, we broke the technology down into the types of databases found, NoSQL and Relational (sometimes referred to as "traditional" SQL). This was done to be able to focus on the specific subsections of the two types of databases when looking through the job postings. These include the overall type, such as NoSQL or SQL, and specific database programs, such as MongoDB or PostgreSQL.

Table 4.1. Types of Databases, NoSQL and Relational.

	Key-Value Store	Document-Store	Column Store	Graph Store	Relational
Pros	-Vertical scalability -Simple	-Good search capability	-Horizontal scalability	-Fast network search -Public linked	-ACID

				data sets	
Cons	-Queries for specific data values -Relationships between data values -Multiple unique keys	-Complex multiple operation transactions -Difficult to design and implement -Incompatible with SQL	-Complex querying -Querying patterns change frequently	-Lack of relationships between data -Poor scalability with large graphs -Specialized query languages	-Poor scalability
Use cases	-User session data -User profiles -User preferences -Shopping cart data	-E-commerce -Content management -Analytics -Blogging	-Content Management -Blogging -Counters -Services that expire -Heavy write requests	-Fraud detection -Graph based search -IT operations -Social networks	
Examples	-DynamoDB -Redis -MemcacheDB -Riak	-DynamoDB -MongoDB -CouchDB -Elasticsearch	-Cassandra -Hadoop Hbase	-Neo4j -ArangoDB -OrientDB	-PostgreSQL -MySQL

Table 4.1 displays the different types of databases listed in Section **2.3.2 Databases**with the pros and cons of each type, use cases currently used by each type of database, and examples of each type that are currently used. These pros and cons helped the team decide which type of database we would use if we needed one within the scope of our project. (Vishwakarma, 2017) & (McCreary, 2014).
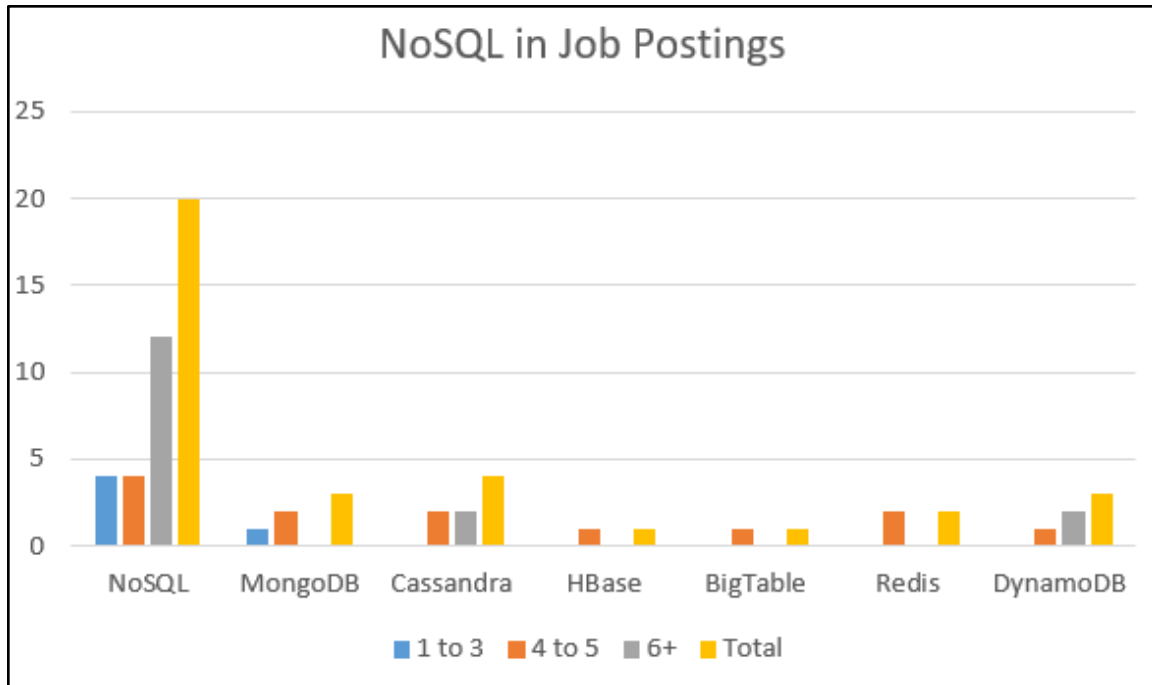
Figure 4.4. NoSQL in Job Postings.

Figure 4.4 displays the amount of job postings that requested NoSQL as a skill along with any specific NoSQL database programs. NoSQL as a whole was requested 20 times (30% of total). Within those job postings, MongoDB and DynamoDB were both requested 3 times (15%), Cassandra was requested 4 times (20%), and HBase and BigTable were both requested 1 time (5%), Redis was requested 2 times (10%).
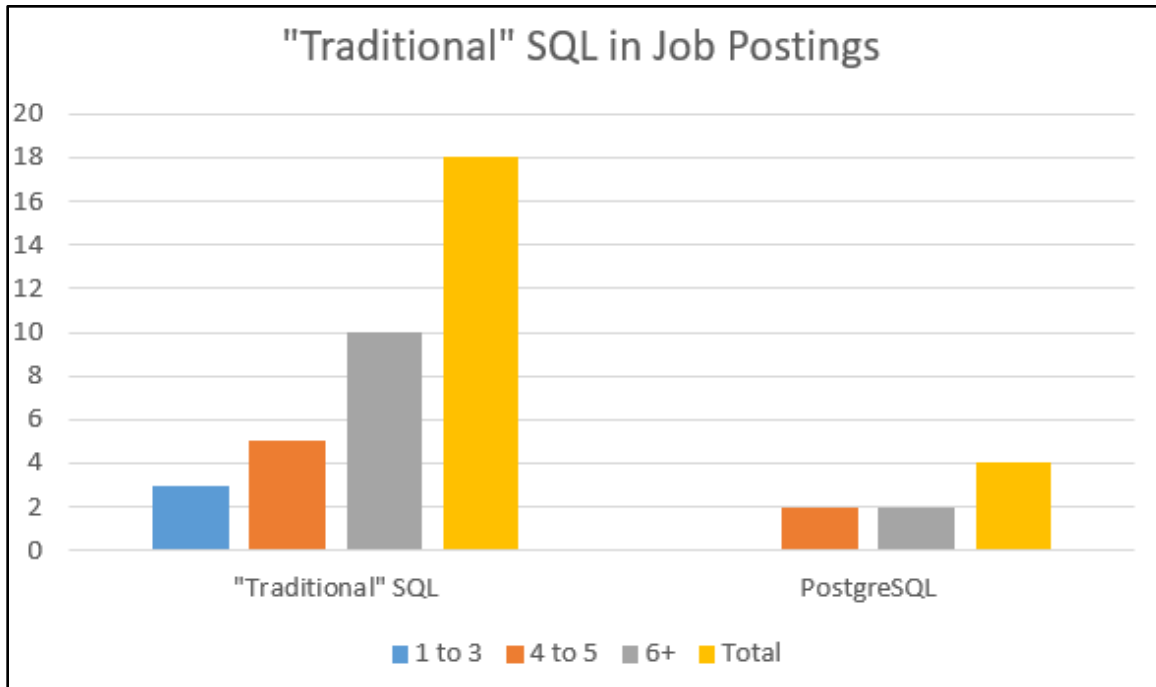
Figure 4.5. Relational SQL in Job Postings.

Figure 4.5 displays the amount of job postings that asked for relational SQL ("traditional") and relational SQL specific database programs. Relational SQL as a whole were requested 18 times (27% of total). Within these postings, PostgreSQL was requested 4 times (22%).

4.4 DevOps

Table 4.2. Types of DevOps Tools.

	Puppet	**Chef**	**Ansible**	**SaltStack**
Pros	-Well developed web UI -Easier to use than pure Ruby -Large community/Good documentation -Comprehensive, even for complex systems	-Great performance -Flexible -Users can share scripts easily	-Supports modules in several languages -Higher security -Simple installation -Lightweight	-Python based -Fewer lines of code (easy to use) -Good performance -Good tools for targeting updates to specific nodes -High scalability with tiered master nodes -Lightweight

Cons	-Must learn Ruby or Ruby-based command language -Difficult installation -Worse performance	-Must learn Ruby -Difficult to use	-Worst performance -Less robust web UI -Scalability issues	-Lack of monitoring capabilities -Less robust web UI
Use Cases	-Developers	-Performance -Developers	-Lack of tiered architecture desired -Sys admins	-Simplicity -Sys admins

Table 4.2 displays the four most requested configuration management tools according to our job posting research. The table includes pros and cons along with use cases for each tool.
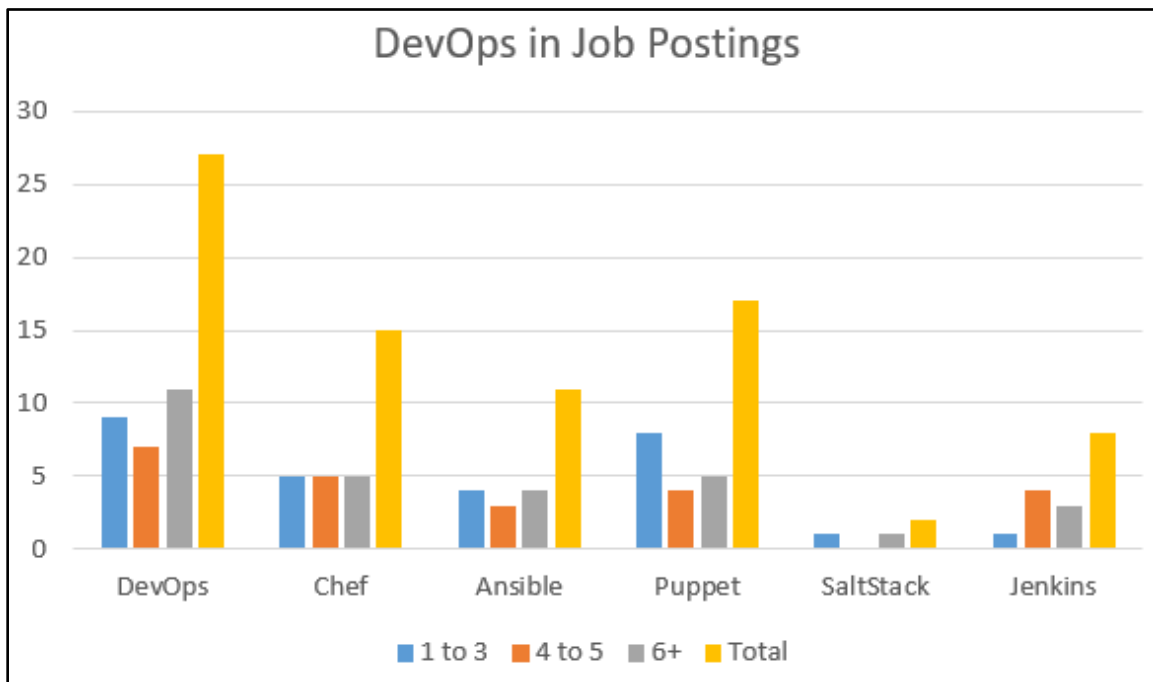

Figure 4.6. DevOps in Job Postings.

Figure 4.6 depicts the number of times the overall DevOps keyword and the most requested DevOps tools were found in our job posting search. DevOps as a whole was requested 27 times (40% of total). Out of those postings, the configuration management tools that were requested were: Chef, Ansible, Puppet, and SaltStack. Chef was requested 15 times (55%), Ansible was requested 11 times (41%), Puppet was requested 17 times (63%), and SaltStack was requested 2 times (7%). Jenkins was the only continuous integration tool that was requested with a total of 8 jobs (30%) listing it as a skill.
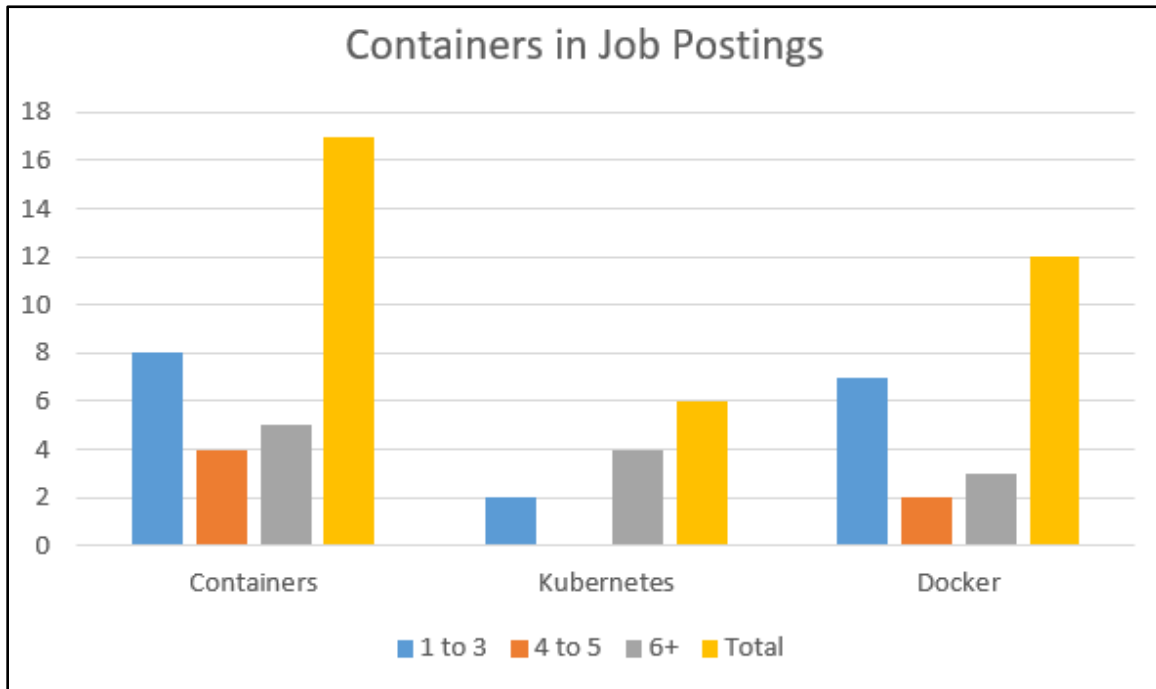
4.5 Containers

Figure 4.7. Containers in Job Postings.

Figure 4.7 depicts the number of times containers overall and specific container tools were mentioned in the job postings we researched. Containers overall were requested 17 times (25% of total). Of these job postings Kubernetes was requested 6 times (35%) and Docker was requested 12 times (70%).
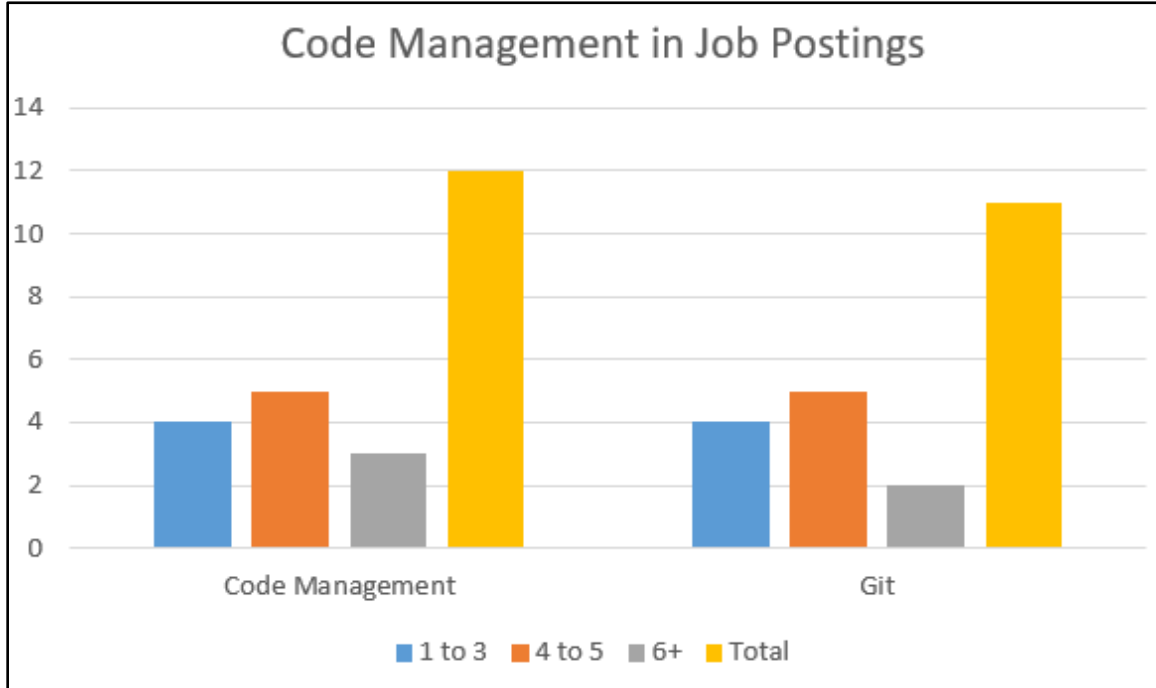
4.6 Code Management

Figure 4.8. Code Management in Job Postings.

Figure 4.8 depicts the amount of requests for code management skills in job postings. 12 job postings (18% of total) requested code management skills. Of these postings, 11 (92%) requested knowledge of Git.
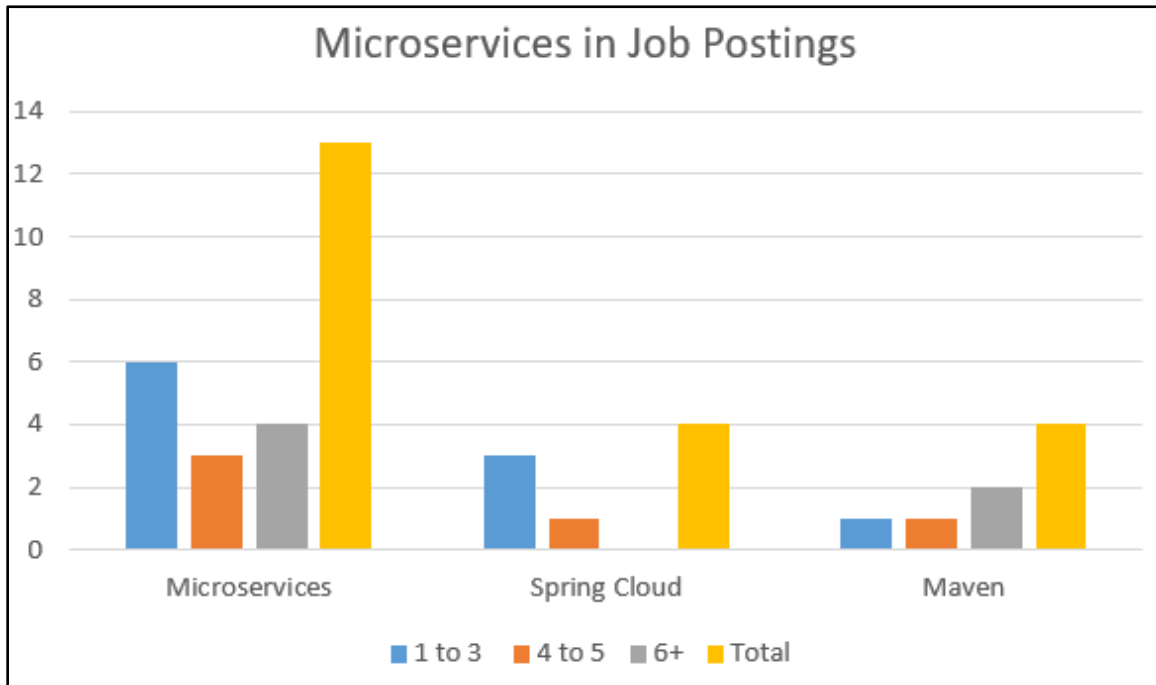
4.7 Microservices

Figure 4.9. Microservices in Job Postings.

Figure 4.9 shows the frequency of microservices and microservice specific tools and frameworks in job postings. Microservices were requested 13 times (19% of total). Within these job postings Spring Cloud and Maven were both requested 4 times (31%).
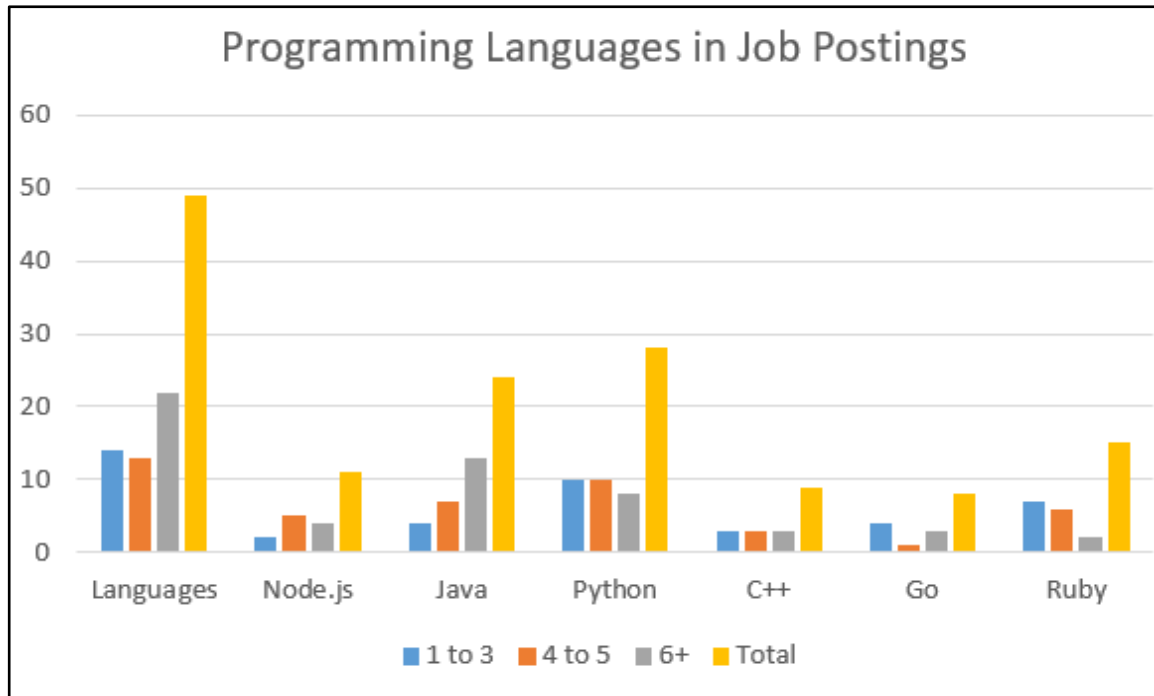
4.8 Programming Languages

Figure 4.9. Programming Languages in Job Postings.

Figure 4.9 depicts the languages requested from job postings. Languages overall were requested 49 times (73%). Within these job postings languages were requested at different frequencies. Python was requested 28 times (57%), Java was requested 24 times (49%), Ruby was requested 15 times (30%), Node.js was requested 11 times (22%), C++ was requested 9 times (18%) and Go was requested 8 times (16%).

4.9 Security

In a 2014 comparative study, different methods of encryption for the cloud were compared to see which covered the most known security issues. The following table is reproduced based on the table in the study.

Table 4.3. Comparison Techniques of Existing Encryption Algorithms (Hemalatha et al, 2014)

Author	Techniques Used	Description	Concepts Used	Security Applied On	Issues Addressed

Padmapriya et al	Inverse Caesar Cipher	Classical Substitution Cipher Same key used for encryption and decryption	ASCII full characters (256 characters)	Cloud customer and cloud provider side	Data security and privacy
Sastry et al	Playfair Cipher	Classical substitution cipher. Same key used for encryption and decryption	5x5 matrix and alphabetic characters used	Cloud customer and cloud provider	Data security and privacy
Maha et al	Fully Homomorphic Encryption	The private key is used for Encryption (without decryption)	Cryptosystem based on fully homomorphic encryption	Cloud provider side only	Data security
Huda et al	Fully Homomorphic Encryption	The private key is used for encryption (without decryption)	Electronic health records classify based on PI (personally identifiable information)	Cloud customer and cloud provider side	Data confidentiality, authentication, availability, and integrity
Sugumaran et al	Block based Symmetric Cryptography	Symmetric layer inserted for encrypting the secure data using a symmetric algorithm	The private key concept is used between sender and receiver	Cloud customer side	Data security and privacy

Monikandan et al	Classical Encryption	Both Substitution and Transposition. Same key used for both encryption and decryption	Plain text is converted to ASCII code value, key range between 1 to 256	Customer's side only	Data security and privacy
Neha Jain et al	DES Algorithm	The same key is used for Encryption and Decryption	Cipher Block chaining mode	Both cloud customer and cloud provider	Data security

Table 4.3 displays the different implementations of encryption that can be used to increase the effectiveness of cloud security. With each different type of encryption, there can be multiple aspects of the cloud industry that can be impacted, depending on the type of security that needed the increase in effectiveness. Security can be tightened on both the consumer and the provider's sides of the cloud, and can tighten just one, or many aspects of the security in the specific cloud instance.

4.10 Company Uses of Cloud Technologies

The list below depicts the results from meetings with Cloud professionals. All the personal identifiers of companies that the professionals interviewed work for have been removed to keep the identities of the individuals interviewed private. We conducted meetings and interviews with industry professionals to gain insight into what companies that deploy onto the cloud or create the cloud platforms use in their workplace.

Cloud Manager, Develops Private In-House Cloud

1. **How does the cloud impact your company?**
   a. The cloud empowers Development teams to deliver code for projects on a timely manner.
   b. Without the infrastructure, setup teams can deliver projects at least 6 weeks ahead of when it would have been delivered before the cloud.
   c. The elasticity of the cloud allows peak seasons to use more resources without having to pay for them for the entire year like before the cloud.
2. **What kinds of cloud technology does your company use?**

a. Kubernetes, for CI/CD: Jenkins. Artifactory for binary artifacts, Hashicorp, Packer, Terraform, Team Management: Confluence, JIRA, BitBucket (code management), Puppet in private cloud.
3. **What skills are you looking for with regards to cloud computing jobs only?**
   a. Out of workforce: broad infrastructure background: networks, Linux, security
   b. Out of school: Development background.
4. **Any skills that you recommend students work on?**
   a. Cloud platform tutorials and books, a student that has worked for six months on tutorials for cloud platforms is worth more than a 15 year developer that has no cloud platform experience.

DevOps Engineers, Develops Cloud Security Software

1. **How does the cloud impact your company?**
   a. The cloud is what the company builds software to protect. Without the cloud they have nothing to protect.
2. **What kinds of cloud technology does your company use?**
   a. AWS Cloud formation, Troposphere (Python), Terraform. CI/CD: Jenkins, Teamcity, Artifactory, codepipeline AWS, Spinnaker, AWS Code Deploy, SaltStack, Chef, Puppet, Ansible, Docker, Kubernetes, Gitlab, Git.
3. **What skills are you looking for with regards to cloud computing jobs only?**
   a. Moving fast, using code auto-deployment. Develop on feature branches. Feature flags (on/off feature switch), unit testing, but automated code and UI testing.

DevOps Engineer, Develops Cloud Security Software

1. **What kinds of project would you suggest having students do to learn cloud technologies?**
   a. Find a quick project that can get an application hosted easily. Put code on Github, use runners to connect and launch the application using Heroku.
   b. Also, have students try to implement new features with little downtime to a running Cloud application. Most end-user products have a 1% downtime that can be used for introducing new features. Giving students a chance to work on a project like that would be helpful.
2. **How did you learn cloud computing technologies?**
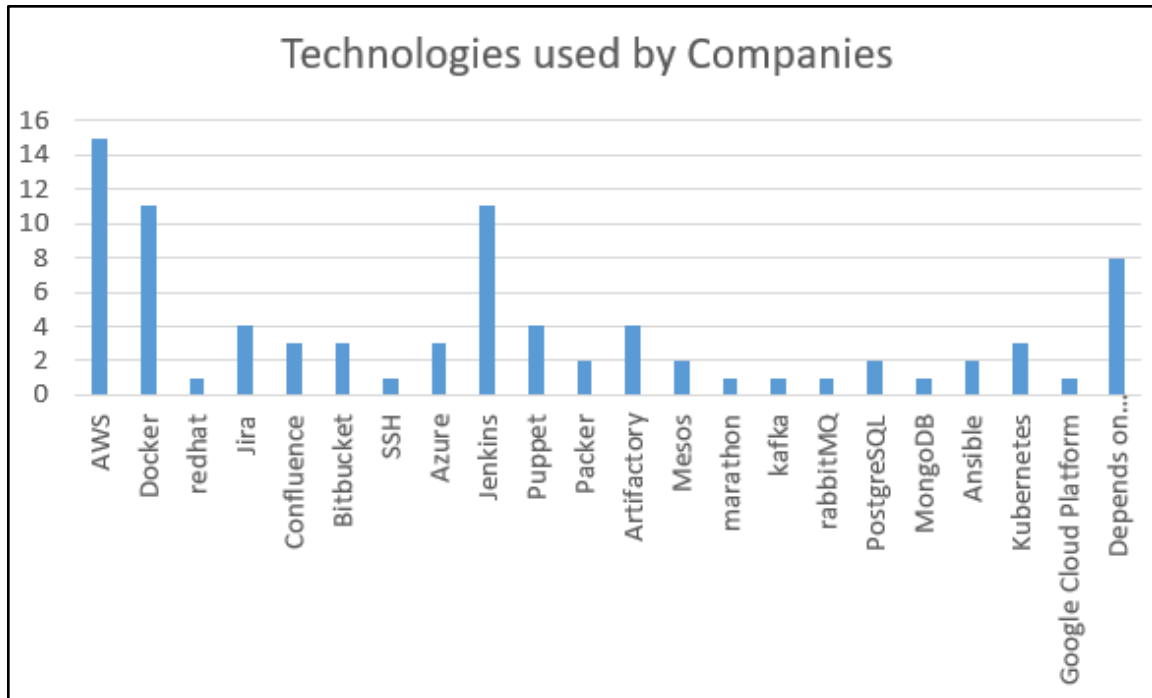   a. AWS on the job, then learned more and more from there.

Figure 4.10. Technology Results from Industry Professionals from the WPI Fall 2017 Career Fair.

Figure 4.10 depicts the technologies companies that attended the WPI Fall 2017 Career Fair use in their development of cloud computing applications. Out of the 21 companies we surveyed during the career fair (10% of total), the following results were given with limited prompting for technologies. AWS was reported to be used by 15 companies (71%). Docker and Jenkins were reported by 11 companies (52%). Eight companies (38%) reported that the technologies depended on the project or team members. Jira, Puppet, Artifactory were reported by 4 companies (19%). Confluence, Bitbucket, Azure, and Kubernetes were reported being used by 3 companies (14%). Mesos, Ansible, Packer, and PostgreSQL were reported to be used by 2 companies (9%). Google Cloud Platform, Redhat, SSH, Marathon, Kafka, rabbitMQ, MongoDB were all reported being used by 1 company (5%).
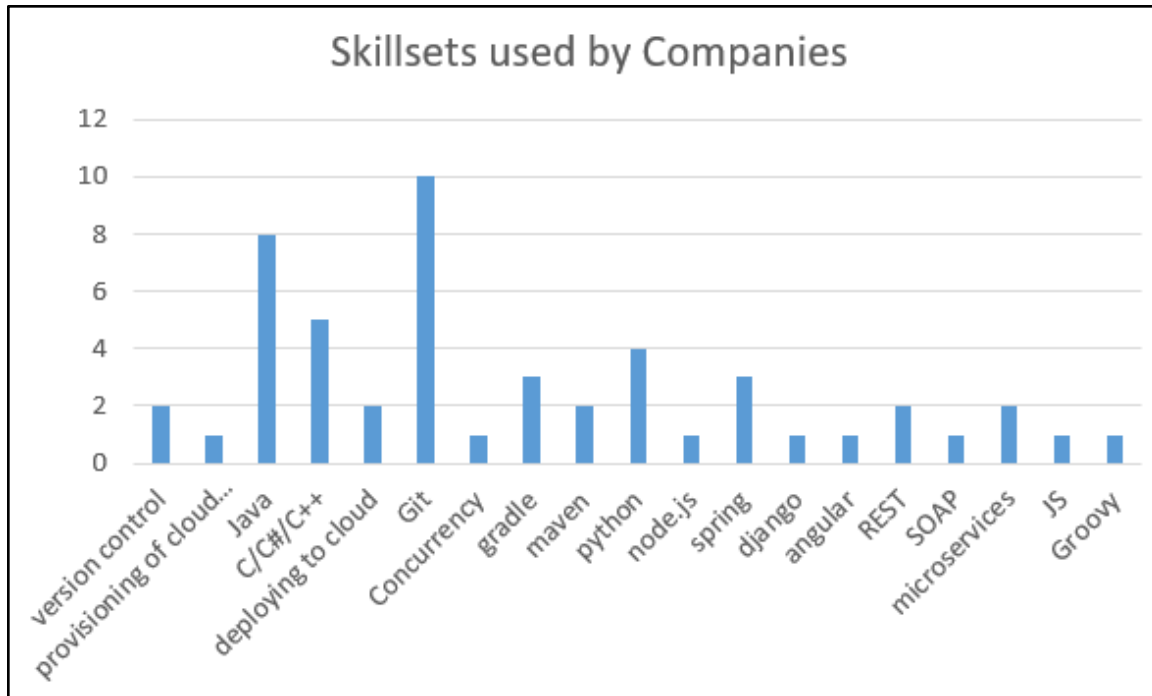
Figure 4.11. Skillset Results from Industry Professionals from the WPI Fall 2017 Career Fair.

Figure 4.11 depicts the skillsets reported by Industry Professionals from the companies we surveyed at the WPI Fall 2017 Career Fair. Out of the 21 companies we surveyed (10% of total) the following were skillsets supplied by the industry professionals with little to no prompting. Knowledge of Git was reported by 10 companies (47%). Knowledge of Java was reported by 8 companies (38%). C/C#/C++ were reported 5 times by companies (24%). Python was reported by 4 companies (19%). Spring methodologies, and Gradle knowledge were reported by 3 companies (14%). Microservices methodologies, version control, REST methodologies, and Maven knowledge were reported by 2 companies (10%). Provisioning of cloud resources, deploying to cloud, concurrency, Django, Angular, SOAP, JS, and Groovy were all reported by 1 company (5%).

5. Discussion

In this chapter, we draw conclusions and recommend technologies. These recommendations are based on research (Section **2. Background**), personal implementation (Section **3. Methodology**), and gathered data (Section **4. Results**). We then discuss limitations of the project.

5.1 Technology Recommendations

In this section we consider the impacts of selecting a technology from each category of our reference model. Ultimately, we provide our recommendations for which technology to choose for particular use cases. The specific tools discussed are accurate as of October, 2017. Though the tools may change, we believe that the underlying concepts will continue to be highly valuable in the future. In this section, "the team" refers to a theoretical software engineering team that wants to implement these technologies.

5.1.1 Platforms

The top platforms are Amazon Web Services (AWS), Microsoft Azure (Azure), and Google Cloud Platform (GCP). While all three were mentioned by companies, AWS showed up far more than Azure and GCP. Based on these results and research, the cloud computing industry appears to be trending towards AWS. The Hadoop framework showed up mainly when researching job postings. Hadoop can be integrated into most platforms as a way to parse big data in and out of the platforms.

Amazon Web Services provides superior free services given to students doing academic work. The free tier itself gives access to many of the services that students would need to implement a basic cloud application for free. AWS's free tier is also good for 12 months, with each month starting a new billing period that resets all amounts used back to zero. As of September 2017, WPI partners with AWS's Educate tier that gives students from WPI $100 worth of credit along with more free tutorials than the free tier provides. One free tier account can grant admin access to multiple users, which is good for academic group project work. One downside is that students need to monitor their accounts carefully, as Amazon will automatically charge for exceeded free tier limits. Additionally, there is no way to open the console of the VM being hosted when hosting a server instance through AWS.

Microsoft Azure's free tier gives $200 worth of credit, but only lasts 30 days, which is less than a WPI term. Azure also does not appear, with in depth research, to offer an education tier like AWS does. Azure does not charge the user unless they subscribe to use services at the end of the trial. From the research done, Azure does appear to allow use of any services it provides. However, the user must use the $200 credit towards provisioning either: space allocated to their account for 14 projects, 40 Databases allocated to their account, or 8 TB of storage. All of these provisions are for the free trial month only.

Google Cloud Platform offers two project environments: the 'standard' and the 'flexible' environments. The 'standard' has a limited free tier that gives $300 worth of free credit, but only for 6 months. There is no automatic charge once the trial is completed. The 'flexible' environment does not have a free tier. GCP does have the ability to open the console for any VMs being hosted. The free tier does have limitations that are not clearly listed out. These limitations can be found on the Cloud platform Console under each individual service. The free tier does supply interactive tutorials, along with documented tutorials. Google Cloud Platform caps the number of project provisions at 12 projects for the free tier. If a user wants additional projects they must request access and pay for the services for that project, even if their trial has not ended.

AWS should be used over Azure and GCP. Many companies and job postings are currently using AWS, and the industry appears to be continuing in that direction. AWS's free tier gives students and individuals working on projects more freedom to learn the technologies than the free tiers from Azure and Google Cloud Platform. The documentation for AWS's services is vast and growing as services continue to be built. The documentation for Azure and GCP are not as detailed and not as extensive. AWS also has the most support for and from other tools and frameworks that will be needed to develop a cloud platform, as can be seen in section 2.

5.1.2 Databases

In the results from the job postings (Figures 4.4 and 4.5), NoSQL and Relational Databases were requested almost equally. As seen in Figure 4.10, the WPI Fall 2017 Career Fair, few companies surveyed reported looking for relational SQL and none reported looking for NoSQL.

Most job postings that asked for NoSQL did not give specific database programs, only requesting that the applicant know how to use this type of database for storage. Most of the job postings that requested knowledge of NoSQL were also looking for six or more years of experience in the workplace, which may suggest that companies do not expect beginner and intermediate skill leveled applicants to have experience using NoSQL databases.

There was only one specifically requested database program that appeared often in job postings and that was PostgreSQL. Over half of the job postings that requested relational SQL were also looking for six or more years of experience in the workplace, which may suggest that most job postings do not expect beginner and intermediate skill leveled applicants to have experience using relational SQL databases.

NoSQL databases should be used over relational SQL databases if scalability is important to the team. If the team wants ACID, then relational SQL should be used. Additionally, relational SQL databases may have better performance for relatively small data sets. For cloud computing, it is likely that the datasets will be large.

If scalability is highly desired, then the different types of NoSQL databases should be considered. These different types are outlined in more detail in Table 4.1. Key-Value Store is ideal for simplicity and vertical scalability. Document Store is ideal for searches. Column Store

is ideal for queries over entire rows. Graph Store is ideal for data that forms a network of relationships. The type of database used depends on the needs for the project, and the needs of the team.

These types of databases do not have dependencies on other technologies, but some platforms include built-in database types that are easy to integrate. For example, AWS offers DynamoDB, a NoSQL database that offers Key-Value Store and Document Store capabilities.

5.1.3 Configuration Management

Most companies that use configuration management tools reported that the tool used was largely personal preference. Teams within companies even used different configuration management tools.

If the team has a complex network of computers, Puppet or Chef should be used. Both tools are comprehensive and offer enough functionality to maintain a system by themselves. Within those choices, unless the team is comfortable with Ruby, Puppet may be preferred because it uses its own simple command language. If the team wants a simple to use tool, then Ansible or SaltStack are preferable. Within those choices, Ansible is quicker and easier to install on a system, but SaltStack is more scalable. See Table 4.2 for a more in depth comparison.

In general, it may be optimal to run two configuration management tools. In this case, the team should choose one of Puppet or Chef and one of Ansible or SaltStack. This approach provides a comprehensive tool as well as a simple and lightweight one.

 In the end, the optimal choice for a configuration management tool is whichever the team is more comfortable using. Each of these four tools have good enough performance and perform the same job function. Furthermore, benefits of using one tool may be negated if the team is unable to configure it properly due to inexperience.

5.1.4 Continuous Integration/Continuous Deployment

During both our research and the results from job postings and the Career Fair, Jenkins was the only Continuous Integration/Continuous Deployment tool mentioned.

Containers

Docker and Kubernetes both work well with AWS, which can explain the frequency of those two tools being requested. Docker should be used by teams that reuse the same libraries and configuration settings across different projects to reduce cross-platform issues. If a team is small, Docker might not be needed.

Kubernetes should be used by a team that has multiple devices using the same containers, such as Docker, to deploy and manage the containers. Kubernetes is not required by teams with a small network of systems or teams that do not use containers.

5.1.5 Code Management

During both the research and the results from job postings and the Career Fair, Git was the only Code Management system mentioned. There were a few different UI tools to implement Git in the workplace. The most common of these tools were Github and BitBucket.

During surveys and meetings with companies, see Figure 4.10, the binary artifact repository tool Artifactory, was mentioned several times. Artifactory can be used for teams that build multiple working versions of their executable files. The tool stores the files in one location allowing for version control of the artifacts. Artifactory should be used if developers outside of the initial team might want to use the binary artifacts in their projects, as the repositories can have different access permissions.

5.1.6 Microservices

For build configuration tools, Gradle should be used. Gradle is open source and builds upon Maven. In a 2017 case study done on the build speeds between Gradle and Maven, Gradle outperformed Maven in every test. The build speed can be equated to developer time, which is money being spent by companies to build projects. Gradle saves more time, thus more money in the long run. (Gradle. Inc., 2017)

Springboot can be used by teams that want to implement the APIs that have been created for Spring. Springboot is not necessary for cloud computing, but will make developing code in Java easier, as many APIs are available for various project features.

All platforms offer built in API gateway services. These services are simple to integrate into their respective platform and should be used to implement microservices with cloud applications.

5.1.7 Language

Specific languages are supported by different platforms. Thus, depending on which platform the team chooses, the language options will change. Once the platform is decided upon, the language selected will vary within those options because, as seen in Figure 4.10, companies use whichever language the team is comfortable with.

Based off our team's implementation of AWS using AWS Lambda and AWS API Gateway, Python is easy to use within AWS. JavaScript was also useful for programming within AWS. From research into the cloud technologies and tools, Ruby can be limited to the development and use of code management tools, such as Puppet and Chef. Java is what many graduates know as it is a well known and used object-oriented language.

Since much of the code is abstracted away by the platform, it is generally best that developers use whichever language with which they are most comfortable. The main consideration for languages should be if there are any custom libraries that would be helpful.

5.2 Project Infrastructure

In this section, we discuss the infrastructure of the prototype application. The application used several AWS services that work together to produce features for the AWS hosted static website. Figure 5.1 depicts the infrastructure the application has to communicate between AWS services and the end users.
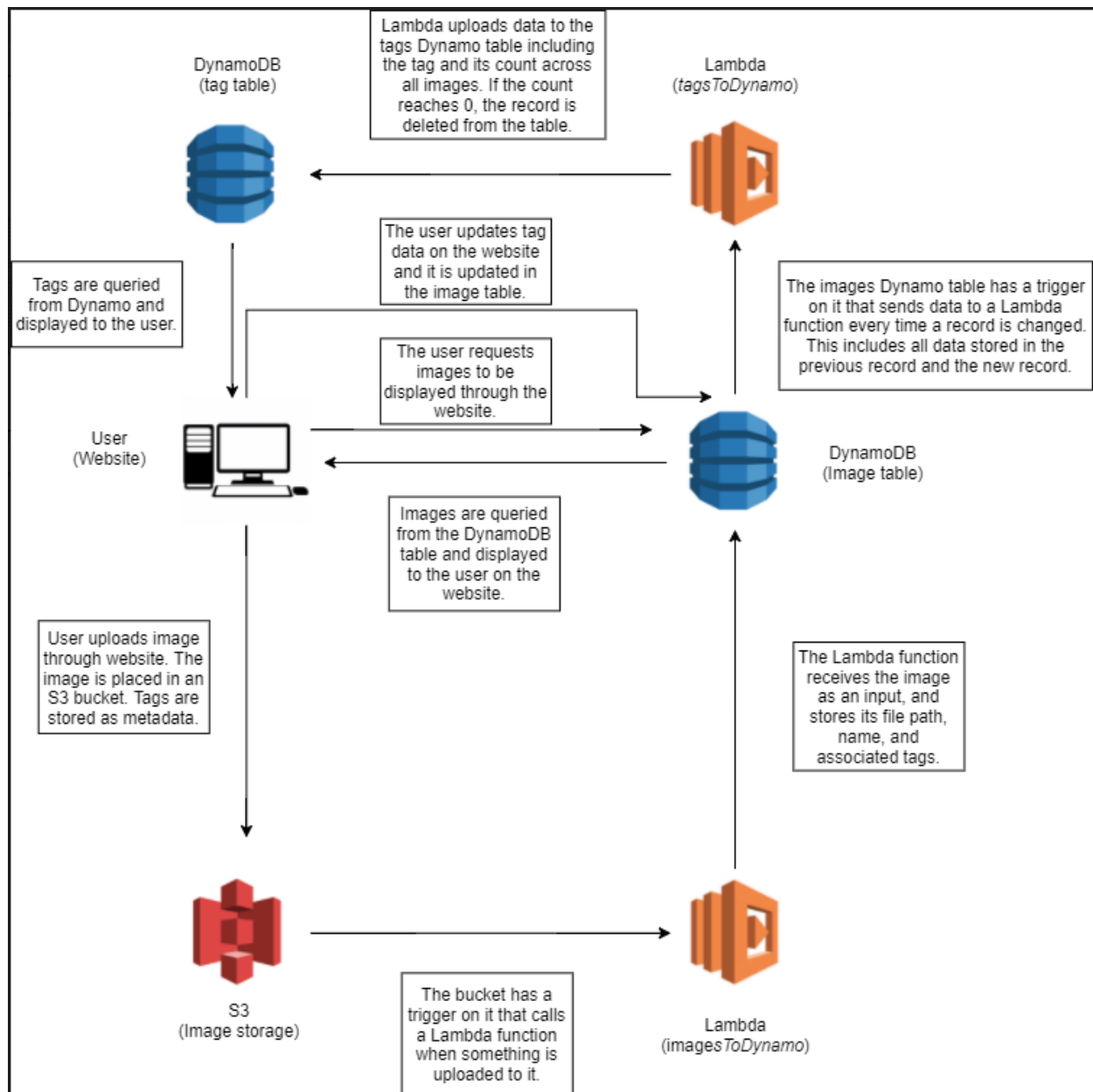

Figure 5.1. Prototype Application infrastructure.

The prototype used two DynamoDB tables, neither of which needed to be set up manually. There were only two Lambda functions, with 190 lines of code. There was one CSS file with 296 lines of code. There were five HTML/Javascript files with 926 lines of code.

The end user starts the application process by requesting one of three actions: uploading an image, requesting images to be displayed via the random image or search features, or updating tag data.

Uploading Images

When uploading an image to S3, the end user invokes Amazon's AWS Javascript API to upload the image to the S3 bucket, which has a Lambda trigger. This trigger calls a Lambda function (*imagesToDynamo*) that places all of the image data (name, uuid, tags, filepath) into the images table in DynamoDB. The image table has a trigger that invokes a Lambda function (*tagsToDynamo*) to store any new and old data each time a table item is changed. The lambda function stores updated data related to tags in a tags table. From there, the database is queried using the AWS API when an end user requests an image, or group of images.

Requesting Images

The end user invokes the AWS API to query directly from the DynamoDB images table. The table then returns the list of images requested by the user. This list is either a list of searched images by tag or a random image depending on if the user searched for tags or requested a random image, respectively. The images are then displayed to the end user on the web UI. Users can also request all currently stored tags by using the AWS API to scan the tags table.

Updating Tag Data

When updating tag data, the end user invokes the AWS API to store the updated information in the images table. This update automatically triggers a Lambda function (*tagsToDynamo*) that stores all updated tags in the tags table. The tags table is then queried to display the updated tags to the user.

5.3 Project Limitations

In this section, we discuss some of the limitations that may have affected the results of our project. It is important to keep these limitations in mind when considering the implications of our research.

The short time frame we had limited the amount data we could gather. Particularly, we were unable to see if trends in technologies and skillsets would have changed several months later. Though this report may serve as an accurate representation of the industry as of October, 2017, the short length means that there may be drastic changes that we will be unable to track. This limitation is especially important to note, as the software industry often goes through rapid change.

With a small team, most of the DevOps tools were not feasible to use. These tools include: Configuration Management tools, Continuous Integration/Continuous Deployment tools, Docker, and Kubernetes. These tools work best for teams larger than most academic course project sizes.

The application implementation was limited to the free tiers of platforms to reduce the amount of money students would have to pay out of pocket during a course at WPI. Thus some services were not used and the free services were called in a limited amount for the application implementation.

6. Conclusion

The project produced three final deliverables; a reference model, a training tutorial, and a prototype cloud computing application. The reference model can be found in Section **3.5 Cloud Application Reference Model**. The training tutorial can be found at the following webpage: http://aws-mqp.blogspot.com/p/overview.html. The prototype application can be read about in more detail in Section **3.6 Image Classification Service** and is hosted at the following webpage: https://s3.amazonaws.com/aws-website-imageclassification-nqrc7/index.html.

The reference model was created as a model for others to build their own cloud computing application based on our market research. The training tutorial was created so that others can recreate the minimum viable product for our prototype application and learn how to use several services offered through Amazon Web Services. The prototype application was created to be able to learn the cloud computing technologies that were researched for this project and to be able to create the training tutorial effectively. The prototype application was an image classification website that allows users to upload, classify, and search for images.

The prototype's infrastructure is broken down into several AWS services: AWS S3, AWS Lambda, AWS DynamoDB, and AWS IAM. All of these services work together, using HTML, CSS, and Javascript, to make the application functional on AWS. The application has only a few hundred lines of code, and uses many templates created by other AWS users, AWS Developers, and outside projects, such as Bootstrap, to limit the amount of code written manually. Section **5.2 Project Infrastructure** has a more detailed description of the infrastructure.

The cloud computing approach to this type of project domain is much simpler than the traditional Java approach. In Java, this application would have required a minimum of thousands of lines of code. Additionally, developers would need to create a domain manually, find the web servers to host the site, and manually write the code for each service AWS provides.

Risk Mitigation

The main risk for the project is potentially being charged for AWS use. Charges may occur if the project exceeds the monthly Free Tier limit or if a service that is used is not part of the Free Tier.

This risk can be mitigated within development teams by switching user accounts if the current user is close to their monthly limit. Any files can be easily swapped over to a different AWS account. Any Lambda functions or DynamoDB tables will need to be recreated under the new account, but it is easy to do so. Users are charged for the use of all services under their account, so if any team members have side projects on AWS, this approach will not work. Users are able to view their current as well as projected resource usage on a monthly basis through AWS's billing dashboard. Automated alerts can be configured to send emails when a certain

threshold has been reached or is projected to be reached. However, it is recommended to closely monitor the billing dashboard, as the alerts may come too late in certain situations.

There is a much greater risk of overusing resources if there are any errors in the Lambda code. If a function fails due to an error, AWS will attempt to call the function again. These retries may repeat infinitely and use up compute resources for each attempt. Additionally, the function may time out if it receives a particularly large request. All subsequent retries will also time out. To remedy time out issues, increase the time out option on the Lambda function. A simple solution to terminate all active Lambda requests is to put a return statement at the top of the function. To prevent additional requests during this time, disable the Lambda function's trigger. It may also be useful to temporarily disable the website. One simple way to do so is to enable the "Requester Pays" option on the S3 bucket hosting the website. If requester pay is not setup, the website will be inaccessible until the option is disabled.

Lambda functions should be monitored closely to avoid any surprise charges. Logs from all Lambda functions can be viewed through CloudWatch. CloudWatch shows if a function terminated early due to an error or time out, and thus will be retried. Logs are stored in streams based on function and updated in real time. A new stream is created for the function each time you push an update to Lambda.

Recommendations for Future Work

Based on our results and information gathered through implementing our application, we would have made the following changes to the project. We would have started with the randomly generated identification numbers for the images to allow for better database table structure. We now know that buckets can only have one trigger of the event type, in our case object PUT. We would have structured our application from the start without planning on having multiple triggers on our S3 bucket. We would have created two database tables from the start for the image and tag data to allow for new features to possibly be developed in the scope of the project. We would have started the user login features at the very beginning of development, since AWS Cognito is a new service without detailed documentation or code templates. We would have used Bootstrap from the start to be able to enhance the UI sooner. DynamoDB uses several reserved words which we used unknowingly, which slowed the development process. DynamoDB does not add triggers immediately, so we would have added them earlier to save development time. We would have ensured that we created all services in the same region to prevent cross-region errors.

For future work on the application, we would develop the features based on the use cases regarding **The Company** and **Moderators**, located in Section **3.6.1 Use Cases**. These use cases were not developed due to time constraints. Some features we would like to highlight from the use cases include deletion of images for Moderators and user accounts that limit the functionality based on user group.

7. Bibliography

About Ruby. (2017). *Ruby - A Programmer's Best Friend.* Retrieved from https://www.ruby-lang.org/en/about/

Amazon Web Services Inc. (2017). *AWS Lambda -Product Description.* Retrieved from https://aws.amazon.com/lambda/details/

Amazon Web Services Inc. (2017). *What is AWS Lambda?* Retrieved from http://docs.aws.amazon.com/lambda/latest/dg/welcome.html

Amazon Web Services Inc. (2017). *What is Cloud Computing?* Retrieved from https://aws.amazon.com/what-is-cloud-computing/

Apache Software Foundation. (2017). *Apache Hadoop Documentation.* Retrieved from http://hadoop.apache.org/docs/current/

Apache Software Foundation. (2017). *Maven Features.* Retrieved from https://maven.apache.org/maven-features.html

Apache Software Foundation. (2017). *Maven Project.* Retrieved from https://maven.apache.org/what-is-maven.html

Asay, M. (2014, March 21). *Google's Go Programming Language: Taking Cloud Development by Storm.* Retrieved from https://readwrite.com/2014/03/21/google-go-golang-programming-language-cloud-development/

Athena Information Solutions (2013, April 1). *A Look at Open Source NoSQL Databases and Cloud Computing*.

Bednarz, A. (2013, September 19). *5 Ways devops can benefit IT; Devops advocates say the IT methodology can improve application quality, speed deployment, and drive revenue.* Retrieved from https://www.networkworld.com/article/2170067/data-breach/data-breach-5-ways-devops-can-benefit-it.html

Benson, J., et al. (2016). *Survey of Automated Software Deployment for Computational and Engineering Research.* Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7490666

Business Collection. (2016, September 23). *Version 2 of the CloudBees Jenkins Platform Debuts*. Retrieved from https://www.highbeam.com/doc/1P2-40065218.html

Chacon, S., et al. (2014). *Pro Git.* Retrieved from https://git-scm.com/book/en/v2

Columbus, L. (2017, April 29). *Roundup of Cloud Computing Forecasts, 2017.* Retrieved from:https://www.forbes.com/sites/louiscolumbus/2017/04/29/roundup-of-cloud-computing-forecasts-2017/#66f49bbc31e8

Crosman, P. (2015, March 26). *'Cloud Containers' May Be a Digital Package Banks Can Accept*. Retrieved from https://www.americanbanker.com/news/cloud-containers-may-be-a-digital-package-banks-can-accept

Database and Network Journal. (2017, February). *Jenkins community surveys, news and developments.* Retrieved from https://www.cloudbees.com/blog/state-jenkins-2016-community-survey-results

Docker Inc. (2017). *What is Docker?* Retrieved from https://www.docker.com/what-docker

Ellis, L. (2014, May 12). *A Brief intro to 'DevOps'.* Retrieved from http://www.translation-please.com/2014/05/

Felipe, E. (2016, December). *Are Microservices the stuff of business application development dreams?.*

Ford, S. (2014, February 17). *What is Artifactory?* Retrieved from http://tech.lifeway.com/2014/01/what-is-artifactory/

Github Inc. (2017). *Bootstrap*. Retrieved September 24, 2017, from https://getbootstrap.com/

Google Inc. (2017). *Google Cloud Platform Documentation.* Retrieved from https://cloud.google.com/docs/

Google Inc. (2017). *Google Cloud Platform Services.* Retrieved from https://cloud.google.com/products/

Gradle Inc. (2017). *Gradle Build Tool.* Retrieved from https://gradle.org/

Gradle Inc. (2017). *Gradle vs Maven: Performance Comparison*. Retrieved September 24, 2017, from https://gradle.org/gradle-vs-maven-performance/

Griffith, E. (2016, May 3). *What is Cloud Computing.* Retrieved from https://www.pcmag.com/article2/0,2817,2372163,00.asp

Hammes, D *et al.* (2014, March 22). *Comparison of NoSQL and SQL Databases in the Cloud*. Retrieved from http://saisconferencemgmt.org/proceedings/2014/HammesEtal.pdf

Heller, M. (2017, August 24). *How devops tools accelerate software delivery.* Retrieved from https://www.infoworld.com/article/3219305/devops/how-devops-tools-accelerate-software-delivery.html

Hemalatha, N., et al. (2014, June). *A Comparative Analysis of Encryption Techniques and Data Security Issues in Cloud Computing, 96*(16)*.* Retrieved from http://research.ijcaonline.org/volume96/number16/pxc3896873.pdf

IBM. (2017). *What is HDFS (Hadoop Distributed File System)?.* Retrieved from https://www.ibm.com/analytics/us/en/technology/hadoop/hdfs/

Knorr, E. (2015, March 2). Walmart Doubles Down on OpenStack. *InfoWorld.com*. Retrieved from https://www.infoworld.com/article/2890873/cloud-computing/walmart-doubles-down-on-openstack.html

JFrog. (2017). *Artifactory - Universal Artifact Repository Manager*. Retrieved, from https://www.jfrog.com/artifactory/

Kozhirbayev, Z. & Sinnott, R. (2017, March). *A performance comparison of container-based technologies for the Cloud.*

Kubernetes Authors. (2017). *Kubernetes Documentation.* Retrieved from https://kubernetes.io/docs/home/

Manufacturing Close-up. (2014, November 25). *Amazon Web Services Unveils AWS Lambda.* Retrieved from http://closeupmedia.com/tech.html

McCreary, D. (2014, November 13). *Making Sense of NoSQL.* Retrieved from http://macc.foxia.com/files/macc/files/macc_mccreary.pdf

Meyers, M. (2012). *CompTIA A+ Certification Exam Guide.* New York City, New York: McGraw Hill.

Microchip releases Industry's First End-To-End Security Solution for IoT Devices Connected to Amazon Web Services' Cloud. (2016, August 11). *Microchip.* Retrieved from

https://www.microchip.com/pressreleasepage/microchip-releases-industry-first-security-solution-amazon-web-services-cloud

Microsoft Corporation. (2017). *Microsoft Azure Documents.* Retrieved from https://docs.microsoft.com/en-us/azure/

Microsoft Corporation. (2017). *What is cloud computing? A beginner's guide.* Retrieved from https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/

Miglierina, M. (2014, September). *Application Deployment and Management in the Cloud.* Retrieved from http://ieeexplore.ieee.org/abstract/document/7034713/

National Institute of Standards and Technology. (2011). *The NIST Definition of Cloud Computing* (Special Publication 800-145). Retrieved from: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Node.js Foundation. (2017). *About Node.js.* Retrieved from https://nodejs.org/en/about/

Pahl, C. (2015, July 15). *Containerization and the PaaS Cloud*. Retrieved from http://ieeexplore.ieee.org/abstract/document/7158965/

Panda, D. (2011, March 20). *Java Cloud Development: What Developers Need to Know.* Retrieved from http://www.developer.com/java/java-cloud-development-what-developers-need-to-know.html

Pavlo, A. & Aslett, M. (2016, June). *What's Really New with NewSQL*? Retrieved from http://dl.acm.org/citation.cfm?id=3003674

Petrillo, F., et al. (2016, September 20). *Are REST APIs for Cloud Computing Well-Designed? An Exploratory Study.* Retrieved from https://link.springer.com/chapter/10.1007/978-3-319-46295-0_10

Pivotal Software. (2017). *Spring Boot Project.* Retrieved from http://projects.spring.io/spring-boot/

Pivotal Software. (2017). *Spring Cloud Project.* Retrieved from http://projects.spring.io/spring-cloud/

Plitchenko, V. (2016, October 06). SaaS, PaaS, and IaaS Explained. Retrieved from https://www.apriorit.com/white-papers/405-saas-iaas-paas

Providers using the Cloud give vendors high marks; non-users are very wary. (2013, August). *Health Management Technology, 34*(8), 7. Retrieved from https://www.healthmgttech.com/

Python Software Foundation. (2017). *General Python FAQ.* Retrieved from https://docs.python.org/3/faq/general.html

Ratan, V. (2017, February 8). *Docker: A Favourite in the DevOps World*. Retrieved from http://opensourceforu.com/2017/02/docker-favourite-devops-world/

REST API Tutorial. (2017). *RestApiTutorial.com.* Retrieved from http://www.restapitutorial.com/

Richardson, C. (2017). *Pattern: Monolithic Architecture.* Retrieved from http://microservices.io/patterns/monolithic.html

Richardson, C. (2017) *Pattern: Microservice Architecture.* Retrieved from http://microservices.io/patterns/microservices.html

Richardson, C. (2017). *Pattern: Service Component Test.* Retrieved from http://microservices.io/patterns/testing/service-component-test.html

Silver, A. (2017, June 1). *Software Simplified*. Retrieved from https://www.nature.com/news/software-simplified-1.22059

Spiegel, R. (2011, December). Ford Uses Cloud for Efficiency. *Design News, 66*(12), 28. Retrieved from https://www.designnews.com/

Torberntsson, K. (2014). *A Study of Configuration Management Systems: Solutions for Deployment and Configuration of Software in Cloud Environment.* Retrieved from http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A732615&dswid=5773

Vishwakarma, R. (2016, May 16). *The Different Types of NoSQL Databases*. Retrieved from http://opensourceforu.com/2017/05/different-types-nosql-databases/

Volz, D. (2016, May 11). House blocks Google-hosted apps, Yahoo mail over security fears. *Reuters.* Retrieved from http://www.reuters.com/article/us-usa-cyber-congress/house-blocks-google-hosted-apps-yahoo-mail-over-security-fears-idUSKCN0Y22QH

Waxer, C. (2015, August). *Promise and Peril in the Journey to Devops.* Retrieved from https://www.computerworld.com/article/2955117/application-development/promise-and-peril-in-the-journey-to-devops.html

What is Broad Network Access? - Definition from Techopedia. (n.d.). Retrieved from https://www.techopedia.com/definition/28785/broad-network-access

What is Measured Service? - Definition from Techopedia. (n.d.). Retrieved from https://www.techopedia.com/definition/14469/measured-service-cloud-computing

What is Rapid Elasticity? - Definition from Techopedia. (n.d.). Retrieved from https://www.techopedia.com/definition/29526/rapid-elasticity

What is Resource Pooling? - Definition from Techopedia. (n.d.). Retrieved from https://www.techopedia.com/definition/29545/resource-pooling

Yegulalp, S. (2017, February 16). *Docker's tops for devops, AWS is the cloud king*. Retrieved from https://www.infoworld.com/article/3170885/cloud-computing/dockers-tops-for-devops-aws-is-the-cloud-king.html

Appendix A

Career Fair Questions

Note: Ask representatives for their role in the company (ie. HR, Manager, Dev) and tailor these questions accordingly. For example, if speaking to a dev, Question 2 would be "What kinds of cloud technology do you work with? Which are the most important?"

Companies **WITH** Cloud Computing

1. How does the cloud impact your company?
2. What kinds of cloud technology does your company use?
3. What skills are you looking for with regards to cloud computing jobs only? Do they happen to overlap with the non-cloud computing jobs?

Companies **WITHOUT** Cloud Computing

1. Does your company have any doubts about the cloud? If so, do you know what they are and why they might be?
2. Does your company have any plans to move towards the cloud? If so, do you know what those might be and when they might take place?
3. Do you use cloud technologies, but from a third party? If so, why?
4. What skills are you looking for regarding the positions you are offering?