Tech Review

Bailey Singleton: Cloud Technician
Group 19: Brewhops
November 2, 2018
Fall 2018
Senior Capstone

Abstract

This document details options considered for storing, hosting, and deploying Ninkasi's data. Ninkasi has an ever-growing supply of data that is taken weekly, and updated often on their web application. We are going to pick an option that is affordable, yet reliable. We have decided that Amazon Web Services and their suite of excellent tools would provide us with a smart set tools that fit our needs.. In correlation with Docker and/or Kubernetes, we can create an easily deployable, highly scalable, modern web application.

I. Introduction

Ninkasi Brewing has time-sensitive data that needs to be viewable from anywhere in the company, anytime. That is why we are going to push their data and to a cloud-based service. Not only will we use a cloud service, but we will containerize their system so it is easy to track and create more instances if necessary. It will make it very easy for us to test our site and spin up multiple instances. This web app will also need somewhere to go, so we will need to pick a hosting option that will give us the services that fit our needs.

II. Containers

A container is a standardized unit of development. They live above the infrastructure and operating system, and encapsulate versions of an application. It keeps all the dependencies and application code in one environment. The name ”Container” is actually a very descriptive name, as they simply ”contain” applications to their own environment. Containers allow us to create instances of the web application easily and quickly. This is useful for testing and deployment, and allows team members make their own changes without impacting each other. Also, deployment options can use these containers to spin up more of them if necessary. Contianters will help move the project forward quickly, and to allow efficient development.

A. Option 1: Docker

Docker is a well-known container service that is secure, multi-platform, and enterprise-grade. It was originally created as an open-source container technology, called Docker Engine, with the idead to extend Linux Containers known as LXC.[1] Each Docker container to follow this idea: it runs on the same kernel, with other resources isolated per container. Since then, it has become full blown Docker, a leading container provider, and is used by many large companies. [2] Not only is Docker a trusted container service, it would be very easy for us to use, as we all have some experience with Docker, allowing little to no overhead on setting up and teaching about Docker. It is available on Windows 10, Linux, and Mac.

B. Option 2: rkt

Another option, rkt (pronounced ”rocket”) is developed by CoreOS, a new company that was founded by a former OSU student. Rkt features a pod-approach to containers, and includes a well-defined pluggable execution environment. It has a surface area that works well with other systems, integrating easily. It has it’s own built-in management system, so we would not need to use something like Kubernetes to manage our containers. This is definitely a bonus, as it could allow us to not have to rely on quite as many services. Containers execute in a shared space, and each ”pod” is easily configureable, and self-contained to it’s own area, while still sharing resources. It is similar to how Unix’s process model works, in that there is no central container source, and that each container can provide all it needs to run. CoreOS was built to be a more secure, interoperable, and open alternative to Docker. [3]

C. Option 3: Kubernetes

Kuberenetes is more than just containers. It builds containers, yes, but also is a container *orchestration* system. It operates at the container level like Docker and rkt, but are not attached to the kernel and libraries. Each instance is decoupled from the underlying infrastructure, which allows them to be portable across cloud providers since there is no hardware associated with any of its containers. They also provide tools to control all containers, making it easy to manage them. [4] Kubernetes would be good to use in correlation with Docker, as Docker is a good tool for making containers, and Kubernetes is a good tool for orchestrating them.

D. Final Recommendation

Docker is the logical choice, as it is a leading provider of containers. It can also be used with Kubernetes, as Kubernetes can orchestrate the containers. This will most likely be the route that is taken with Ninkasi Brewing.

III. Cloud Service

Cloud services make it easy to keep all data centralized, but not so centralized that it is at risk of being exposed or crashing. Keeping data locally can pose issues of losing it all in the case of an accident. The main focus of choosing a cloud provider will be the ease-of-use it provides, reliability, and cost. The cheaper we can get, the better.

A. Option 1: Amazon Web Services

Amazon Web Services (AWS) is one of the largest cloud providers, and is trusted by over a million customers. The product we would find ourselves using most would be Amazon S3. It is an object storage option built to store data and retrieve any amounts of it from anywhere. Amazon boasts it has ”99.999999999% durability” [5] and that is one of the biggest promises we are looking for. It runs on the worlds largest cloud infrastructure, allowing them to offer competitive prices while still providing first-class security. While we are not sure exactly how much storage we will need, their pricing is for Standard Storage is $0.0023 per gigabyte, monthly. We might be able to get away with infrequent storage, as data will be uploaded about once a week. This costs only $0.0125 per gigabyte.

B. Option 2: Google Cloud Storage

Google Cloud Platform seems very similar to Amazon Web services in durability. They boast the same 99.999999999%. [6] They provided the same amounts of availability as well, so it really comes down to cost. The prices per gigabyte that Google provides is between $0.02-$0.035 per month. So, overall, Amazon Web Services would be the better choice, over Google Cloud Platform.

C. Option 3: Microsoft Azure

At this point, all cloud platform providers are the same in terms of storage. They all offer some form of security, reliability, and durability. Azure provides about the same standards of it’s competitors, but it is the most expensive of the three options. For storage, it clocks in at about $0.06 per gigabyte. this would make it the least likely for us to use in the long run. Microsoft does not boast about their durability percentages like the others, so I cannot relay the information. This would be our last choice.

D. Final Recommendation

Amazon Web Services has the most customers, and is the cheapest, making it an easy choice for the project. If Google Cloud Storage turns out to be a better use in some way or another, the project could switch to using their products instead.

IV. Web Hosting

Web hosting will be an important option for our product as it comes time for production. Currently it is being hosted on an OSU student’s engineering domain and this won’t last. A web hosting site that strikes a perfect balance between reliability and affordability will be the most enticing.

A. Option 1: Amazon Web Services EC2

With AWS web hosting, we can use Amazon Web Services S3 combined with their hosting option. This way we would only have to deal with one company for all of our website needs, using Amazon EC2 in combination with S3. They are fully elastic, allowing us to increase our web size in minutes, instead of having to wait hours or days. we can have many servers running simultaneously. Pricing will be a factor here, and there are several different types of EC2 instances. We will need at least one server running all the time, so we would probably like to go with a dedicated or reserved instance. These cost somewhere between $0.05 to $0.096 per vCPU-Hour. [7]

B. Option 2: Heroku

Heroku uses their own branded Dyno for servers. They are basically just a server like the rest, but offer lightweight features that allow for cheap serving of websites. A simple web server on Heroku comes to about $25 per month. This is cheaper than Amazon’s EC2, which is about $36 per month. This option makes more sense based on price alone. They are used by a large assortment of companies, such as Citrix, Macy‘s, and many others. [8]

C. Option 3: GoDaddy

GoDaddy is by the far the most barebones on this list. They offer the cheapest pricing, but at a performance and ethical cost. Since we will be using a database, it is important to look at how much space they give us. GoDaddy only offers 1 gigabyte of storage, and that definitely will not be enough space. The $1 per month price tag is very enticing, but its shortcomings will not be enough for us to even consider. Not only that, but their reputation as an ethical company seemed to end at it’s conception. The site is overrun with ads, and let us not forget the ad campaigns that were very demeaning and sexist towards women. While they offer a free URL, [9] everything else the company stands for is not something we would like to support.

D. Final Recommendation

First and foremost, GoDaddy is the the least-recommended of the list. Amazon Web Services is the best choice, if they were to be used in conjunction with their cloud storage. This would give Ninkasi a centralized location for all storage and hosting needs. If a different storage option is chosen, then Heroku would make for a great fit.

V. Conclusion

Our options for cloud computing come as a bundle. From the options provided, it is easy to see that if we were to pick an option from one category, it would make the most sense to take a similar option (if not the same company) from another category. Overall, the best bang-for-buck while still maintaining excellent performance would be with Amazon Web Services, using Docker containers to contain our applications. Amazon Web Services is the cheapest option for a reliable storage and hosting, with easy scalability and little to no downtime. We could use Docker and Kubernetes to orchestrate containers as well, since Amazon offers an Elastic Container Service that would fit our needs perfectly. [10] This would put all of our options in one place, making it easy for developers and our clients to manage.

References

[1] Docker. (2018) About docker engine. https://docs.docker.com/engine.
[2] ——. (2018) Docker customers. https://docs.docker.com/customers.
[3] UpGuard. (2018) https://www.upguard.com/articles/docker-vs-coreos.
[4] Kubernetes. (2018) What is kubernetes? https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/.
[5] Amazon. (2018) Aws customer success. https://aws.amazon.com/solutions/case-studies.
[6] Google. (2018) Storage products. https://cloud.google.com/storage/features/.
[7] Amazon. (2018) Amazon ec2 reserved instances pricing. https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/.
[8] Heroku. (2018) Customer success. https://www.heroku.com/customers.
[9] GoDaddy. (2018) Godaddy. https://www.godaddy.com.
[10] Amazon. (2018) Amazon elastic container service for kubernetes. https://aws.amazon.com/eks/.