AWS: Aligning Digital Infrastructure for Efficient Development

CS4991 Capstone Report, 2022

Nicholas O'Connor
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
nbo5td@virginia.edu

ABSTRACT

Amazon employs an extraordinary number of software developers. Having so many programmers, manually allocating resources, especially computational infrastructure, in response to project needs is difficult, if not impossible. To solve this issue, Amazon utilizes dynamic allocation to bring resources online only when needed. By breaking apart the different aspects of software, such as HTTPS responses, databases, and analytics, the resources required for each can be handled without needing to account for how other aspects handle theirs. This allows for teams of programmers to operate in a similar manner, often needing no knowledge of internal workings at all in order to work together efficiently. Eventually, this system could be simplified and expanded with external user stories for use by non-programmers.

1 INTRODUCTION

Imagine multiple painters trying to work on the same canvas, using the same paint and each needing access to a full set of tools. How do you coordinate such that two brushes aren't painting over the same spot at the same time? Is there a way to avoid bringing out one large supply of paint at once? Instead of giving each artist a complete set of bushes, can their use be coordinated such that brushes are shared and no artist needs to wait their turn? These concepts, of utilizing limited shared development space, and limited resources, are important to software developers who aim to keep costs and production time down. When working on a codebase as a team, it takes more than just clear communication to ensure developers don't step on each other's toes. Similarly, processing power and network availability comes at a cost. Figuring out how to turn off resources when they are not needed reduces power consumption, both in terms of computation and electricity.

2 RELATED WORKS

The research and motivation behind solving this problem has been covered by multiple authorities. In particular, Amazon's AWS whitepaper enumerates a number of advantages that cloud computing carries when it comes to pooling shared resources. A few selected advantages include:

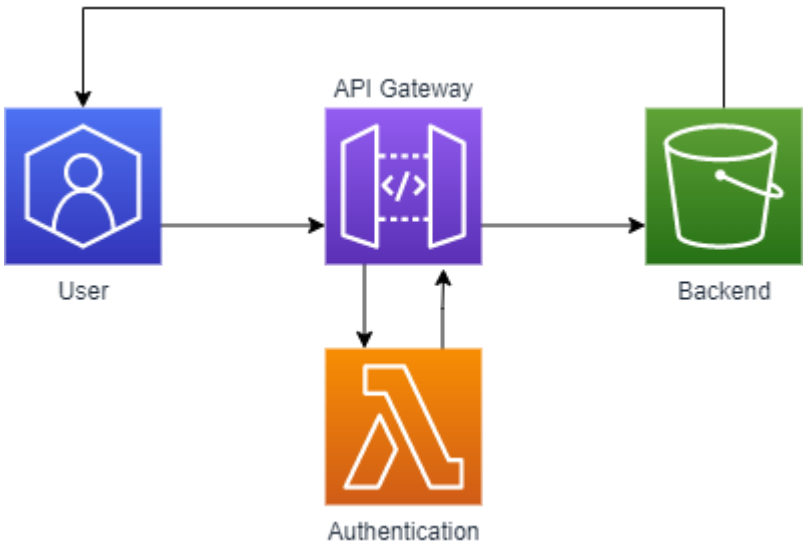

Figure 1: Essential Design Structure

- Larger fixed expenses can be traded for variable, per-demand expense
- Benefits from an economy of scale
- Easily expandable capacity
- Less money spent on maintenance

All of these contributed to the design and capabilities of my internship project. [1]

3 SYSTEM DESIGN

My project was broken down into two major parts: redesign and update the frontend user interface, and provide a method of automatic authentication through Amazon's internal security service. Included in the second objective was ensuring that any future development would continue to validate all traffic for proper authorization.

3.1 Frontend Interface

For the web application that the end user would interact with, there was already an older version of the widget available that I could draw requirements from, in addition to the ones provided by my project lead. My task was to update it with a cleaner look, as well as streamline user activity.

For the library used to build the application, I chose React. There were many advantages to using React over other frameworks; my main reasons I decided on it were previous familiarity, trust from React being an open-source library, as well as wide use within Amazon for other applications. For the visual framework, I selected one that was available internally to Amazon projects, and allowed for a consistent look across all Amazon applications. The framework also had a large support base within Amazon, which made it useful for learning its uses and getting help when needed.

3.2 Authorization Interceptor

In order to ensure that all data being sent to and from the backend was being accessed by an authorized user, I had to incorporate architecture that would intercept traffic and check it for authentication. The authentication also needed to be done in such a way that any developers who would pick up my project in the future, either to work on the backend data handling or the frontend web

application, could do so with the knowledge that all accesses would be authenticated.

To accomplish this, I designed a system which is illustrated in the diagram in Figure 1. Whenever the user interacts with the web application to send or request data, that request is first "caught" by the API Gateway. It sends the request to the authenticator (represented by AWS Lambda), which has access to Amazon's internal security system to check for a valid token. If the user is not signed in and/or authorized, the authenticator returns the login webpage, and prevents the request from reaching the backend. If the user is signed in and has proper authorization, the request is forwarded normally to the backend, which then returns or processes the data without needing the use of either the API Gateway or authenticator.

3.3 Challenges

One of the biggest challenges I encountered was implementing the authentication interceptor. The package I needed to use in order to interface with Amazon's security system was written in JavaScript; however, the rest of my project could only be completed using Java. This caused a language conflict that had to be solved using a workaround. After a couple of days of reading documentation, I contacted more senior developers for help. They were able to aid in adjusting compilation parameters so that I could instead write my project in JavaScript.

4 RESULTS

The finished result of my internship project was a backend framework for a currency conversion tool that supported automatic detection and login of users on Amazon's VPN. It was set up in such a way that all data or traffic sent from the frontend widget to the backend database was intercepted by an authenticator to determine if the user had permission to access the backend services in the first place. The system was also designed such that any future development on either the front user interface, or the backend data management, would not interfere with the interceptor, and all future data would continue to be checked for authorization.

My internship project also included a rudimentary user interface for viewing transaction rates. This interface used an updated framework over a previously used widget, and was intended as a direct update. It included cleaner design and a more streamlined decision process for the user, as well as a sidebar menu so that all of the commonly-used features would always be accessible. Initial tests suggested that users would be able to complete their tasks approximately 10% faster.

4 CONCLUSION

Overall, this project provided both an exceptional learning experience, and a solid foundation for future work on the application. I now feel more confident using cloud computing services for developing web-based applications, as well as more familiar with AWS services specifically. My project gives structure for future work to build upon. The use of cloud-based services ensures that it is not using excessive computational power, and that it is flexible for future development to maintain and improve it.

5 FUTURE WORK

While my project provided a decent base structure for the intended application, more work is needed to fully implement the business logic behind handling user data. I anticipate that much of the future development will be using similar infrastructure, i.e., AWS products and services. I will also be returning to Amazon to pursue full-time employment after graduation, so I may actually be able to continue working on the same project.

REFERENCES

[1] 2022. Overview of Amazon Web Services. Retrieved February 22, 2022 from https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html