Deployment of Web Applications in the AWS Environment

CS4991 Capstone Report, 2023

Alan Gray
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
apg2fds@virginia.edu

ABSTRACT
Over the course of the last three summers as an intern for QinetiQ, a DC based defense contractor, I added features to existing projects and developed new tools as needed. I used the React framework, deployed with various Amazon Web Services (AWS). I designed the solutions to use a React front end for easy changes and add-ons using AWS Amplify to deploy the front end, and a combination of AWS Cognito, API gateway, and DynamoDB in order to sign in, and retrieve and store data. Finally, I used AWS lambda functions to tie everything together. This experience allowed me to learn a lot about AWS and full stack development, while producing useful products in a workplace. I found that the currently available cloud computing options make standing up new projects very easy for full stack developers, allowing us to focus efforts on more important parts of the project. In the future, I anticipate adding additional security features to lock down the applications more.

1. INTRODUCTION

To understand how web applications work, there are a few key concepts to know: the front end, the back end, and how they are connected. The front end is how the user interacts with the application, which in web development is usually some code that produces a web page (i.e. a combination of HTML, CSS, and JavaScript).

I wrote my front end code in the react framework, which uses an extension of JavaScript called JSX to produce HTML, and JavaScript and CSS for styling. I developed the back end for these projects using AWS's DynamoDB service, which just stores key-value pairs like a JSON file. However, because it is hosted through AWS, it allows for autoscaling as necessary to keep accessing data from the front end fast. In this project, I connected the front and back ends by using AWS API Gateway and AWS Lambda functions. Through those, I essentially made links that, when accessed, returned some data to the front end.

2. RELATED WORKS

Most directly-related works that I took inspiration from are under NDA and cannot be discussed here. However, the main source that made it possible for me to design these sites was the Udemy React course (Grider, 2016). This is an excellent course for anyone seeking to learn more about the React framework. It assumes students know very little and carries them through to a moderate level of competence.

In addition, for anyone seeking to learn more about how AWS services work, there is a 13-hour YouTube video that goes through everything needed to pass the AWS cloud practitioner exam and teaches everything one may want to know (Brown, 2021). For those just looking to produce something similar to the project described here, the whole course may not be necessary, though shorter segments may be helpful.

3. PROJECT DESIGN

This project description covers all the services and connections necessary to deploy web applications to AWS, starting with the frontend, then the backend, then the middleware used to connect the two.

3.1 Frontend

For the projects I did with QinetiQ, the front ends were all developed in React. React allows for a combination of HTML, CSS and JavaScript to be served relatively easily to the end user by using a Model View Controller design. From there, I gave AWS amplify access to the GitHub repository that stores the React Code. AWS Amplify is a nice solution in comparison to running this on EC2 instances because besides providing a small file that tells Amplify how to run the code, there is very little overhead to be done. This allowed me not to worry about the EC2 instance going down, since AWS uses cloud services to keep Amplify instances up all the time.

3.2 Backend

The backend for my projects was hosted in the form of a DynamoDB instance that allows for NoSQL storage of the data. This enabled quick deployment and more additions to the database without having to worry about a schema, since the users of the sites I developed would not be inputting data directly into the database, allowing me to use a more relaxed schema.

3.3 Middleware

The middleware for this project is really what tied it all together. For authentication into the app, I used AWS Cognito, which allows for easy authentication within the React framework. It also allows for account detail verification through post sign up scripts, meaning I was able to verify that any emails provided were in QinetiQ's domain, and then verify the user had access to that email through email verification rather than allowing for malicious users to enter any email on that domain to gain access to the website.

The scripts used for verification with AWS Cognito were AWS Lambda scripts, and they also provided much of the middleware that the user did not directly see. The other major script beyond the account verification was a Lambda script that accessed data from the database when the user needed it.

To organize and better serve those AWS Lambda scripts to the webpage in a more readable format (rather than some excessively long default Lambda script URL), I used AWS API Gateway. This also allowed me to control access to the data much better, including the ability to shut down access to the data should that be necessary, for a database refactoring or something similar.

4. RESULTS

Overall, the technology stack described here worked very well for me. I was able to use the same stack to stand up two different projects during the summer, and it is now my go-to tech stack when I need to set something up in AWS. It provides a cloud solution that is rarely offline, while not costing a lot of money to host because of the relatively simple architecture. It also provides an easy way to share changes in close to real time when I was trying to iterate on the sites' designs with someone and could not share my screen for local testing for whatever reason.

5. CONCLUSION

Overall, this project provides a good basic framework for others to start with when dealing with AWS. This can act similarly to as a skeleton solution, enabling someone new to AWS to easily deploy a web application. This project is more of a quick start guide for understanding the basics of deploying, rather than an extensive list of everything one will ever need to know regarding AWS.

6. FUTURE WORK

In terms of future additions to my projects with QinetiQ, I intend to add a head site allowing access to a portfolio of the products I have developed for the company. A single URL would provide easier access for those who are helping me, and could make it easier for less technical people inside the company to get some examples of what we can do.

In addition, this project did not go into domain registration, which is something I intend on implementing in the future. This would allow the sites to have URLs that make sense instead of the URLs automatically generated by AWS lambda, which tend to be long , random combinations of letters and numbers. This step would be necessary to roll these types of projects out to the end customers eventually anyway, so getting that running properly before rolling it out would be helpful. AWS, which has this service built in, allows clients to buy domains and deploy various sites within that domain to prevent spending more money on multiple domains.

7. ACKNOWLEDGMENTS

My mentor and supervisor for this project was Marcellus Black, a UVA graduate that works at QinetiQ. He helped me find the right solutions for problems as they arose and helped me iterate on the UI design of the sites.

REFERENCES

Grider, S. (2016). *The complete react native + hooks course | Udemy*. The Complete React Native + Hooks Course. https://www.udemy.com/course/the-complete-react-native-and-redux-course/

Brown, Andrew. (2021). *AWS Certified Cloud Practitioner Certification Course (CLF-C01) - Pass the Exam!* [Video]. YouTube. https://www.youtube.com/watch?v=SOTamWNgDKc.