AWS Cloud: How Web Applications Can Improve Product Usability

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Aidan Ricci

Spring, 2023

Technical Project Team Members

Aidan Ricci

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Rosanne Vrugtman, Department of Computer Science

AWS Cloud: How Web Applications Can Improve Product Usability

CS4991 Capstone Report, 2022

Aidan Ricci
Computer Science
The University of Virginia
School of Engineering and Applied Science Charlottesville, Virginia USA
aiqr25@gmail.com

Abstract

A large cloud service provider recognized that its genomics analysis solution was difficult for less tech-savvy genomics scientists to use. As a result, they decided that a web application for their command line interface (CLI) service could reduce the learning curve of joining the service, ultimately bringing in more researchers. My team and I had to create an application programming interface (API) to convert the command lines to usable responses which could be given to the visual web application. We put the API and the user interface into docker containers and wrote cloud development kit (CDK) scripts to automatically provision resources and start the app on the cloud without any manual setup work. By the end of the summer, the web app and API had the ability to run a genomics analysis workflow from start to finish using the same commands the CLI used on the backend. Because a similar service was being developed elsewhere in the company we decided that our project was slightly redundant. If the service continued to be developed, next steps would include improving the visuals and user experience of the web application, adding functionality with additional cloud services (e.g. Uploading workflow files to an S3 bucket for the user), and writing out the rest of the genomics CLI commands into the API.

1. Introduction

How can a genomics researcher conduct research without needing to buy and set up high performance computing hardware? The best solution to this problem is cloud services, which can provide the same functionality as on-premises hardware without any of the upfront cost or set up requirements. Despite services being available in the cloud today, genomics researchers do not widely use the cloud to run their DNA/RNA sequencing workloads. This happens for a number of reasons, but one contributing factor is the usability and technical knowledge requirements of some of these services. Therefore, the purpose of my summer project was to offer a service which could bring new researchers into the cloud for their genomics workloads by reducing the amount of technical knowledge needed in order to begin. This meant taking the existing Genomics CLI, which required a lot of technical knowledge (Writing YAML files, using a command line tool, knowing cloud services, etc.) and turning it into a web app to be used as a management

console. This would allow users to jump straight into genomics workloads in the cloud, with little prior knowledge and set up.

2. Related Works

[1]Undated documentation for Amazon's Genomics CLI project provided inspiration for my design of a similar website. In particular, the Amazon project informed the structure for the API that I had to build out on the backend.

[2]September 2022 documentation for Docker provided help in solving coding issues for my project since Docker was used for making the frontend and backend containers in my project.

[3]The Nextflow website is the home site for one of the DNA Sequencing workflow engines which would be used in the Genomics CLI. I used this site and the documentation to better understand the workflows I was developing my program for to inform my development.

3. Project Design

This section covers the design of my internship project. It starts with an overview of my system architecture and what technology was used to build the project, followed by a section going over the requirements for the project, and finishing with the challenges I faced creating the project.

3.1 Review of System Architecture

The project I created was essentially a full stack web application developed with Python and Javascript. The frontend used Javascript's NextJS framework and MaterialUI to bootstrap the User Interface (UI). It connects to the backend using Javascripts fetch method to grab information from the backend's Application Programming Interface (API). The backend of my application used Python's Flask framework to make an API out of the Genomics Command Line Interface (CLI). The Genomics CLI was installed on the server and python code was used to call the command line commands within the program and then convert the responses into usable JSON data for the API. The API would then simply send the responses to the frontend as requested.

I decided to put the frontend and backend into separate Docker containers in order to give them consistent environments for development and production. It also made it easy to set them up within containers in the cloud as well. Finally, the entire website had to be set up on the cloud automatically, so cloud development kit (CDK) scripts had to be made so that users could simply run the script on the cloud and have the entire website set up for them.

3.2 Website Requirements

The website created for this internship project was intended to function like a management console for the Genomics CLI. As a result, by the end of my internship the website had to run a DNA sequencing workload on the AWS cloud without the use of the command line. This meant the majority of the CLI commands would need to be implemented and there would need to be ways to click through them on the website. The website would also need to be set and run from within the cloud easily. This led to CDK scripts being developed and the addition of Docker containers for the front and backends. For

the sake of the proof of concept, logins, organizational structure, and other useful AWS services like S3 were removed. The UI was allowed to be barebones for the MVP, so there were little visual requirements.

3.3 Challenges

There were a number of major challenges during the development of this project. One of these obstacles was the use of environment variables to store the API keys and domain names. It worked locally, but when the program went on the cloud my environment variables would disappear. The CDK script also had to add the location of the domain of the API on build time of the environment in the cloud which ended up taking a week. Another challenge was the time constraints on the project. Due to other requirements of the internship, I did not get to start the project until halfway through my internship, and had to finish it two weeks early. This gave me a little over a month to write the website.

Finally, near the end of the project the code pipeline used for updating the project broke, breaking the CloudFormation stacks which had existed for over a month. As a result, I had to take a few days near the deadline for the project to simply fix the CloudFormation stacks and the code pipeline so that I could test my code online.

4. Results

The outcome of my efforts on my internship project was a completed website which used the Genomics CLI to run an example DNA sequencing workload from the beginning to the end. This reduced the learning curve of joining the genomics service by removing the CLI knowledge, YAML file writing, and much of the required cloud knowledge. The goal was to bring more researchers to the cloud genomics service.

However, over the course of the summer my team became aware of a full new service in development within the company which could fulfill the same goal. This led support for my internship project to stop, and it converted my project from a potentially usable project into a learning experience and a way of evaluating my internship. As a result, the project scope was reduced and an additional project presentation and new service FAQ document were added to my project as materials for evaluation. This was done both to evaluate me and to give me the experience of working on a new project within the company. The result of the new scope of my project was extra experience in writing and public speaking, which would be useful in my role as a Solutions Architect. This project helped me to secure a return offer from the company for next year.

5. Conclusion:

My internship project from this past summer made it easy for a larger number of genomics researchers to pick up and use a genomics tool in the cloud. The genomics tool, which started as a command line interface service for running DNA sequencing workloads in the cloud. Although already useful, that genomics tool was improved through my work by enhancing the usability of the tool. The usability was boosted by providing the genomics service a website as a graphical interface in addition to reducing the required cloud knowledge and removing the required YAML file writing. Reducing the learning

curve helps to move genomics researchers to the cloud, which in turn will provide them the provisioning advantages of the cloud. These advantages include not having to run their own hardware or guess at the computing capacity needed. Other advantages included having a large amount of built in security, along with the cost benefits of massive economies of scale.

6. Future Work:

There is no planned next phase for this project, which was scrapped because development of a similar service within the company created redundancy. However, if one wanted to continue this project next steps would include improving the visuals and user experience of the web application, adding functionality with additional cloud services, and writing out the rest of the genomics CLI commands into the API. Adding functionality with other AWS services could include uploading workflow files to an S3 bucket for the user and adding authentication, among other things.

7. UVA Evaluation:

The UVA CS program has been extremely useful in preparing me for a software development job; however it has had a few noticeable weaknesses in my experience. The first problem I had was that I took the pilot program, which means I took the major data structures and algorithms as a first year, leading to some degradation in my knowledge by the time I was being tested for jobs and internships. DSA1 and 2 were both extremely useful classes that taught me all the concepts I needed to know for these coding tests.

Although seemingly not necessary for a basic Software engineering job, COA1 and 2 as well as Operating Systems taught me a lot about computers at a basic level, which has helped me to write better code and understand the systems I work on. However, I have found that SDE did not do a great job teaching me the team and testing concepts that could be useful on the job. Also, discrete math and theory of computation have not proved to be useful to me in a CS career, although they did marginally improve the way I think about developing my programs.

Finally, electives have been exceedingly useful in filling out my knowledge of computer science, giving me unique skills which have expanded my skill set and opened up more options for careers. As a whole the UVA CS program has been stellar for preparing me for a basic software engineering role, from interviewing for the job up to writing my own code, leaving a bit of a shortage on debugging and team work. The UVA Computer Science electives have also expanded my horizons and allowed me to think about moving into other roles like Machine Learning and Cybersecurity though I did not gain enough knowledge in any of these areas to get a job without a significant effort on my own.

References:

[1]Anon. AWS Genomics CLI Documentation. Retrieved September 23, 2022 from https://aws.github.io/amazon-genomics-cli/docs/

[2]Anon. 2022. Docker documentation. (September 2022). Retrieved September 23, 2022 from https://docs.docker.com/

[3]Related Anon. Nextflow. Retrieved September 23, 2022 from https://www.nextflow.io/