Deploying a Food Blog Web-Application using Amazon Web Services

Brad Cooley, Sagar Prabhu, Bhakti Patrawala

1. Introduction

Technology is transforming the way we communicate knowledge and thoughts, from print media to digital platforms. One of the most common methods is to use blogging platforms, which provide an engaging and dynamic style that may attract a larger audience than traditional media. We set out to build a web application for blogging purposes using Amazon Web Services and its various managed services for our project. Our goal is to create a scalable, low-latency web application that provides a positive user experience.

To make this a reality, we utilized a range of powerful technologies, including React as the frontend framework and Node.js and Express as the backend. We followed an iterative phase deployment of the application in the cloud using various Amazon Web services like Simple Storage Service (S3), Elastic Beanstalk, CodePipelines, and Relational Database Service (RDS) with a MySQL instance. By combining these services, we created a highly scalable and low-latency application that could deliver a highly enjoyable user experience. For example, Simple Storage Service (S3) provides an incredibly durable and cost-effective way to store and retrieve data, while Elastic Beanstalk automates the deployment and scaling of our application. CodePipelines streamlines our software release process, and Relational Database Service (RDS) with a MySQL instance provides a robust and scalable database solution.

Overall, our web application offers a powerful, dynamic, and versatile platform for sharing information and ideas with a wider audience. Whether you're an individual blogger, a media company, or a business looking to engage with customers, our solution can help you achieve your goals. We believe that our project will contribute to the evolution of digital media and provide a valuable tool for bloggers and content creators.

2. Background and Related Work

Blogging has become an integral part of modern digital communication, offering individuals and businesses a powerful platform to share their thoughts, ideas, and expertise with a global audience. However, building a user-friendly, content-rich blog application can be challenging, especially with the rapidly evolving technologies and changing user expectations. To address this challenge, we aim to create a blog application that leverages the latest technologies and tools to develop a cutting-edge blog application that is both scalable and efficient.

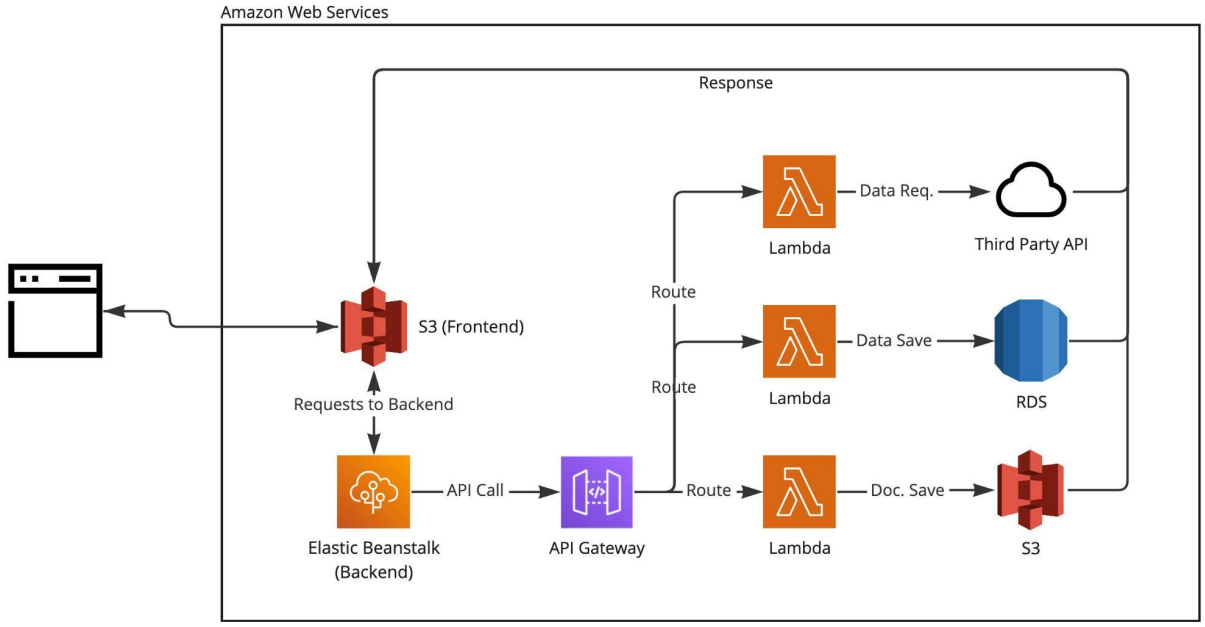
Numerous blogging platforms are available today, ranging from free, open-source solutions to high-end commercial products. Some of the popular platforms include WordPress, Blogger, Medium,

and Tumblr, each with its unique features, advantages, and drawbacks. WordPress, for instance, is a widely used content management system that offers a wide range of plugins, themes, and customization options, making it ideal for bloggers and businesses of all sizes. Similarly, Blogger is a free blogging platform that offers easy setup, intuitive interface, and integration with Google services.

However, with the growing popularity of React, Node.js, and AWS, there is a significant demand for custom blog applications that can leverage the power of these technologies.
React, Node.js, and AWS services like Elastic Beanstalk, API Gateway, RDS, and S3 are all used in our blog web application to create a user-friendly, content-rich blog application. By utilizing the adaptability and scalability of these technologies, we hope to develop a blog application that can provide a seamless user experience, high performance, and effective data management. Overall, our suggested blog application strives to provide a cutting-edge, adaptable, and effective platform for businesses and bloggers to share their ideas, knowledge, and skills with the world.

3. Proposed Project Design & Architecture

3.1 Cloud Architecture



For our application, we planned to utilize Amazon Web Services (AWS) and many of its managed services. In the following subsections, we will describe each managed service and how we intended to use it, the associated costs with using it, and how it aided in the overall concept of cloud computing.

3.1.1 Simple Storage Service (S3)

Simple Storage Service, or S3 for short, would be the holding place for the frontend of our website. We planned on statically serving the frontend and then linking it to Elastic Beanstalk for all API calls. The other purpose of S3 was for document storage for any photos or other things that might come with blog posts. S3 can store a decent amount of data before you are charged for it, so we didn't anticipate exceeding the free-tier limits.

3.1.2 Elastic Beanstalk

Elastic Beanstalk serves the purpose of being a fully managed service that deals with horizontally and vertically scaling virtual machine instances. We planned to use it specifically, focusing on the deployment and availability of our backend. The reason we projected that only the backend would need to be deployed is that React apps can be served statically from an S3 bucket (as mentioned above). Elastic Beanstalk, like most elements we chose in the architecture diagram, is free-tier eligible. Again, we didn't envision meeting the threshold for requests or scaling resources, so there would be no cost.

3.1.3 API Gateway

API gateway allowed for the development and routing of RESTful APIs. API Gateway also allowed for the routing of traffic when a request is made to an endpoint. It integrates well with other AWS services, like Lambda functions, to carry out requests. We planned to use API Gateway for the routing purpose as it managed scaling up and down endpoints (although our endpoints are automatically scaled because they are Lambda functions). API Gateway allows for 1 million API calls under the free tier, a threshold we didn't come close to hitting.

3.1.4 Lambda

Lambda functions are serverless, stateless compute that are critical in an event-driven programming paradigm. We planned to use Lambda functions to serve API requests as they, theoretically, can scale infinitely. Lambda functions are available under the free tier, and we didn't hit any thresholds to be charged for invocations of the Lambda functions.

3.1.5 Relational Database Service (RDS)

RDS is a fully-managed database service that can support many querying languages such as MySQL and MongoDB. RDS allowed us to choose our favorite language without worrying about normal DB administration tasks like scaling, authentication/authorization, and server provisioning. RDS was available for free tier users, and we didn't hit any thresholds.

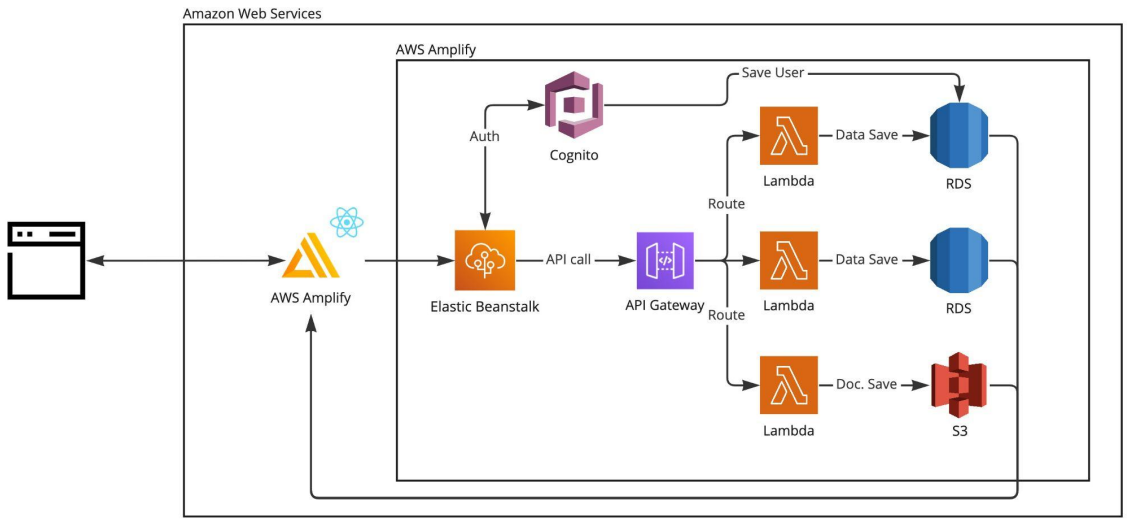
3.1.6 Cost

We believed we could make our application work entirely on the free tier of AWS, but in the event that we reached thresholds, here are some projected monthly costs for the services we chose to use:

Service	Estimated Daily Cost (USD)	Estimated Monthly Cost (USD)
S3	0.007	0.21
EBS	0.41	12.32
API Gateway	0.003	0.10
Lambda	0.00	0.00
RDS	0.00	0.00

Total	0.42	12.63

3.1.7 Researched Alternative Approaches



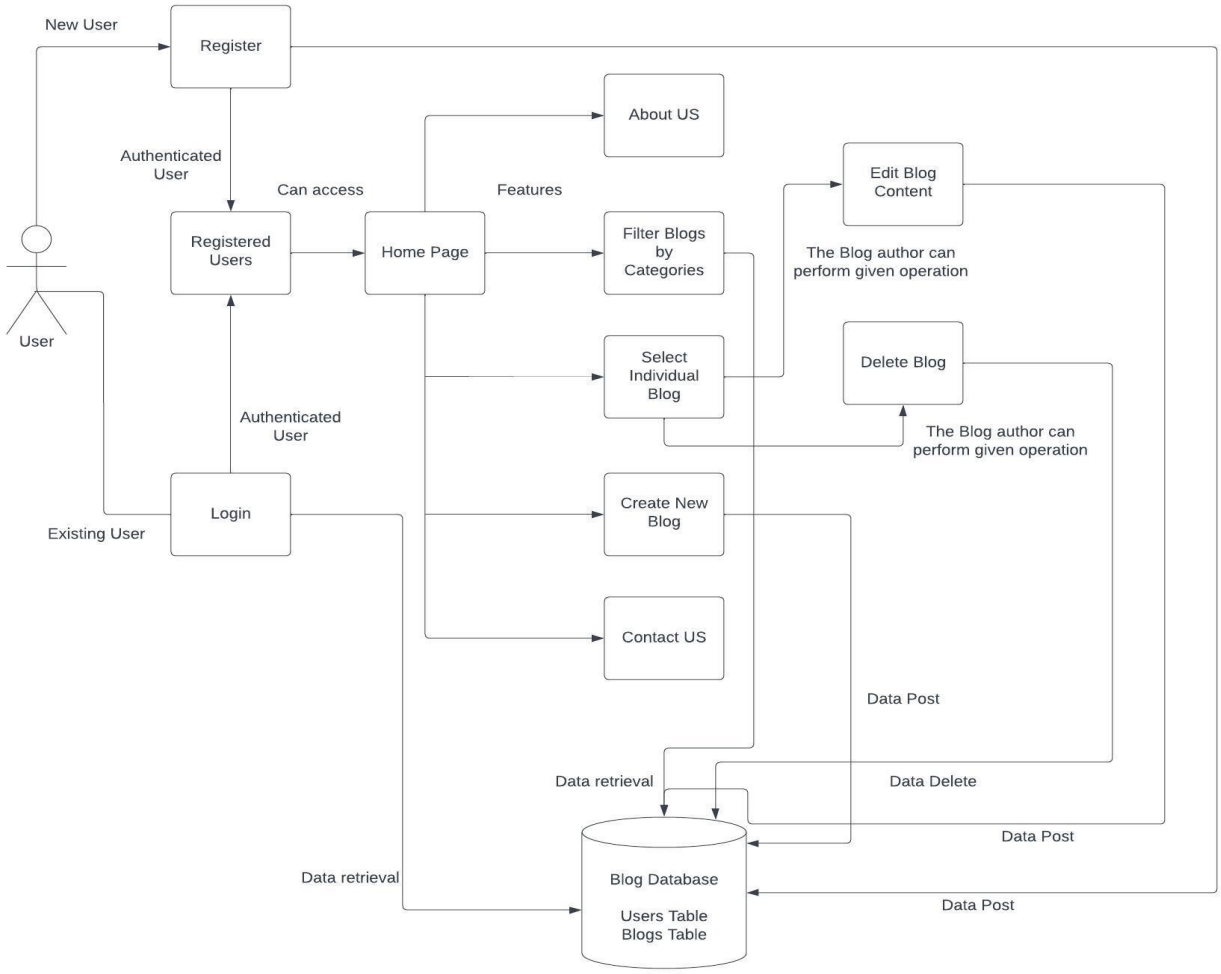
Throughout the research process, we came across multiple other ways to create our cloud architecture and we wanted to highlight one of those here. We didn't use this architecture, but thought it was important to showcase the iteration process we went through in deciding our final architecture.

This architecture uses AWS Amplify, which according to Amazon's documentation, is "a complete solution that lets frontend web…developers easily build, ship, and host full-stack applications on AWS." The difference in this architecture, as compared to ours, is the fact that Amplify manages all these services and the creation of them behind the scenes; all they show to the developers of the application is a nice UI that aids in the creation of all the services. Amplify does a phenomenal job at removing various Infrastructure as Code (IaC) overhead that comes with configuring and managing other AWS services and allows the developers to focus on the buildout and architecture of the application they are deploying.

The other service we are choosing not to use is AWS Cognito, which manages the authentication of users. We are choosing not to use Cognito in favor of other forms of authentication (like Sign In with Google, Sign In with Apple, etc.). Not choosing Cognito allows the platform not to be tied to Amazon for one of the more significant aspects of the application.

3.2 Application Architecture

3.2.1 Application Flow Diagram



The main features of the blog application include enabling users to add and append their own blog articles as well as read blog entries from other users after logging in to the website. Prior to signing in to the application, new users must be able to register themselves. With regard to these features, we have developed the primary web pages so that both the front end and the back end are operational. We have also made sure that our application's end-to-end workflow is uninterrupted from the perspective of a new user. In this instance, we have established the workflow from beginning to end as follows:

- When a new user first arrives at the website, they can browse the user-posted blogs, go through the About Us and Contact Us sections, and register before logging in.
- In addition to being able to log in to the program, an existing user will be able to carry out all the same tasks as the new user.
- Users of the blog application will be able to write and publish their own blogs to the website after logging in and viewing content posted by other users.
- Regardless of whether they are logged in to the site or not, users will be able to contact blog owners via the Contact Us page.
- Users can browse through blog posts that are organized by the categories given to the posts when they were created.

The following pages were our goal when developing our blog application.

1. **Home Page -** Whether or not the user is logged in, this is the initial page that is displayed to them when they open the blog app. The user can access more pages from this one, including those for logging in, registering, writing blogs, learning more about us, and getting in touch.
2. **Login Page -** Users can connect in to the blog application and create their own blogs by entering their login and password.
3. **Register Page -** By providing information such their name, email address, and password, new users can register.
4. **Categories -** When a user selects the categories option from the navigation bar, they can browse postings from various categories.
5. **Blog Page -** When users click on a specific blog post, they will be taken to this page. On this page, users can read the blog post. By clicking on the, users can also modify or remove blogs they've made on this page.
6. **Write Page -** Through this page, users can create new blog articles or edit ones that have already been published. Additionally, they can upload images and choose categories for their posts on this page.
7. **Contact Us -** Users can send an email to the blog's owner using this page by providing their name, email address, and the message they wish to send.
8. **About Us -** General information about our blog application is provided on this page.

The home page has a header, footer, and navigation bar that point visitors to a web page for a different category. Each item card is a representation of a blog post that the site administrator has added. We may examine a longer description of the blog item by clicking on the item cards. This description will include information like a general description of the blog item, related images, and a list of blogs in the same category as the blog we have just clicked on. An anchor to the home page, a categories part that will take us to the web pages of the many categories of the blog, a contact us section, and a about us section make up the left navigation bar.

3.2.2 Categories Page

Users can select the category to which they want to navigate by clicking on the "Categories" part of the left navigation bar. This page will include each individual blog post organized by category in addition to the header, footer, and left navigation bar. Users can access the summary of each blog article and the associated image by clicking on each item, much like on the home page. They can click on any of these specific blog posts, which will take them to the products page. The numerous categories will also be located in the left navigation bar and will be added by the blog application administrator. Users can view the relevant blog posts by clicking on any of these categories.

3.2.3 Blog Items

Users can access the individual blog item page by clicking on any particular blog post. The three most crucial tasks of this page are to show the blog post made by the administrator, delete blog items by clicking the delete icon if the post was authored by them, and allow users to alter articles that they have written. In addition to viewing other entries in the same category as the current blog article, visitors can browse to them as well.

4. Evaluation & Result Analysis

This section will outline the results and changes in both the application and the cloud architecture along with some analyses of the various aspects of the application.

4.1 Application Results

From the application development point of view, we have developed the Register Page, Login, Create/ Update Blog section, View individual blogs page, About US and Contact US page. This section includes the screenshots of the final application and the various pages.

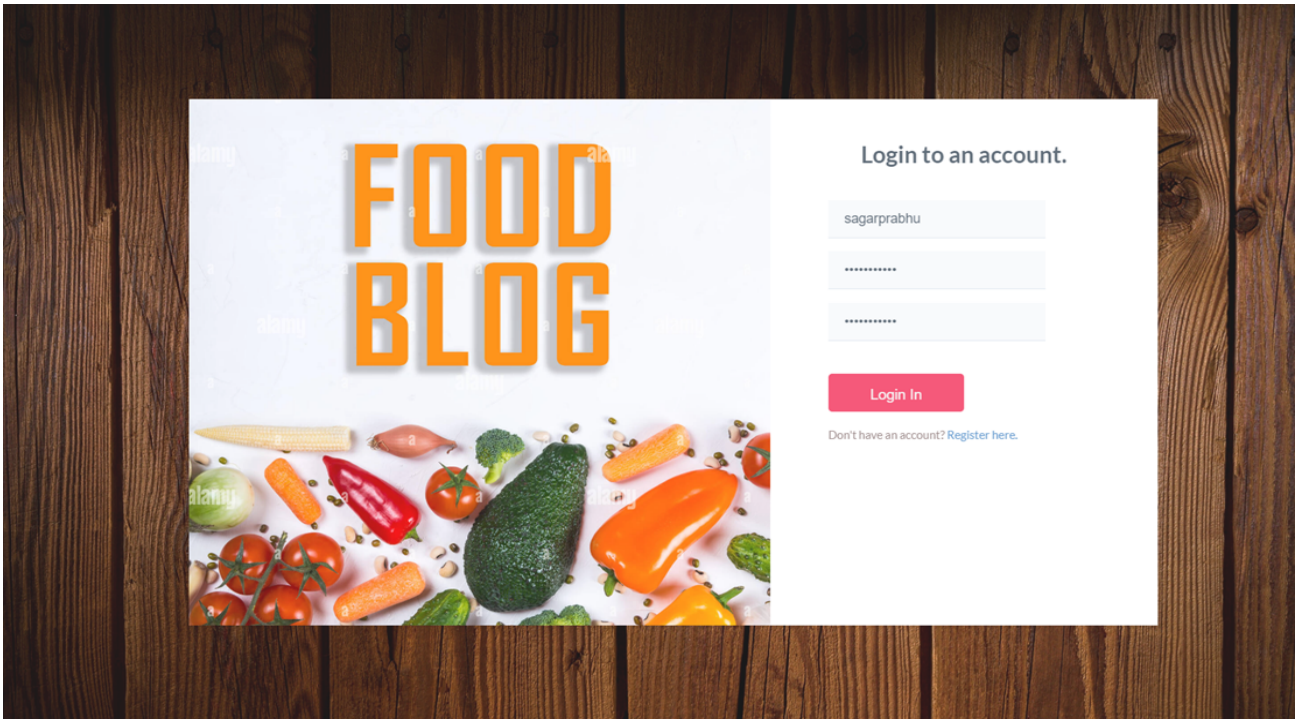
4.1.1 Register Page

The register page enables new users to register to our website entering details like username, email, and a password which will be encrypted and all these details will be added in our Users table in the RDS-MySQL Instance database as displayed below:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Ce

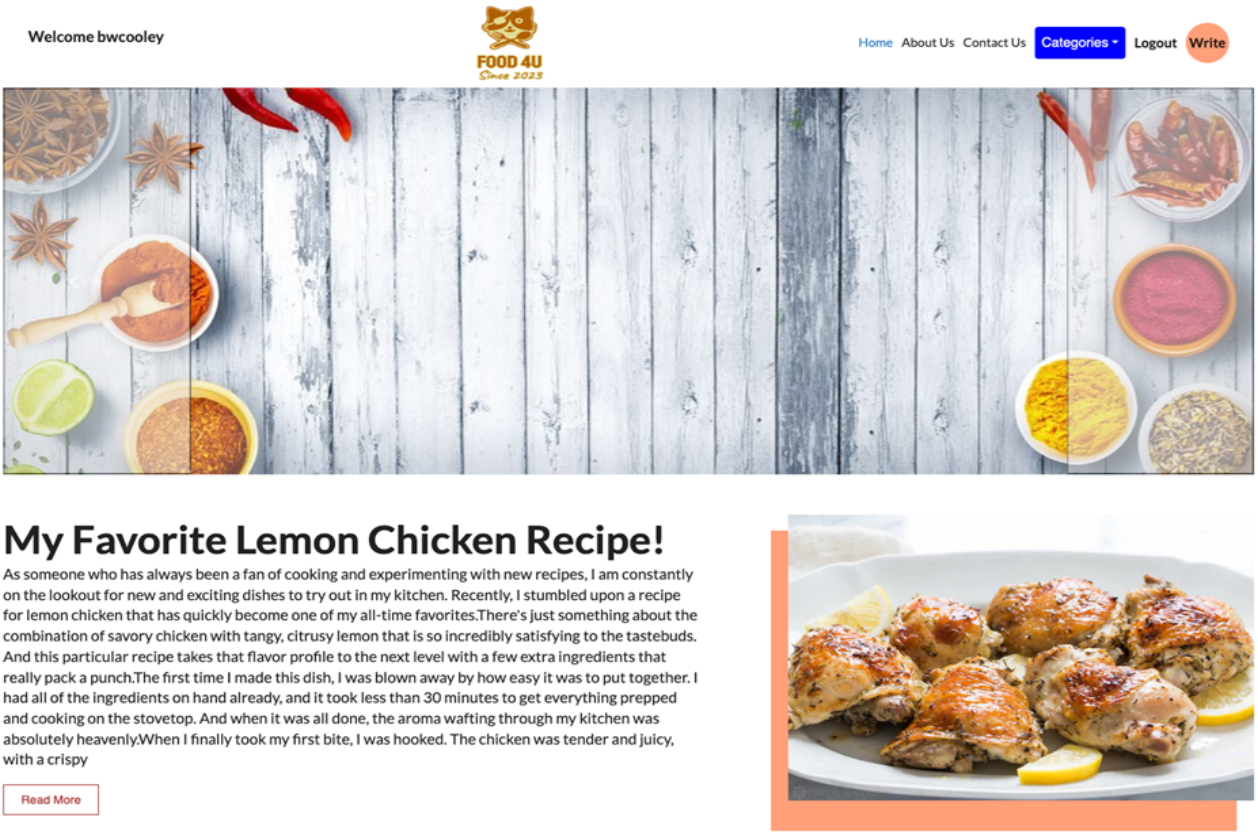
id	username	email	password	img
1	test123	test@test.com	$2a$10$mI12o7wfoqLn70Rl7srTZ.VljeuKUQH3m...	NULL
2	bcooley	brad@cooleyweb.org	$2a$10$ffNn8gkcyRNQAqZb3FDiUu7TVkv8liRw...	NULL
3	sagarprabhu	sagarprabhu@gmail.com	$2a$10$fcq3L4kHzE/bS7k47saELeU4WCgo7dpq...	NULL

4.1.2 Login Page



Users can log in using login credentials like username, and password and enjoy access to the Food Blog website. They can view the blog and create new blogs or manage the existing blog details. If any user has an application query or any issues, they can access the Contact Us page and send us a mail along with the query.

4.1.3 Home Page

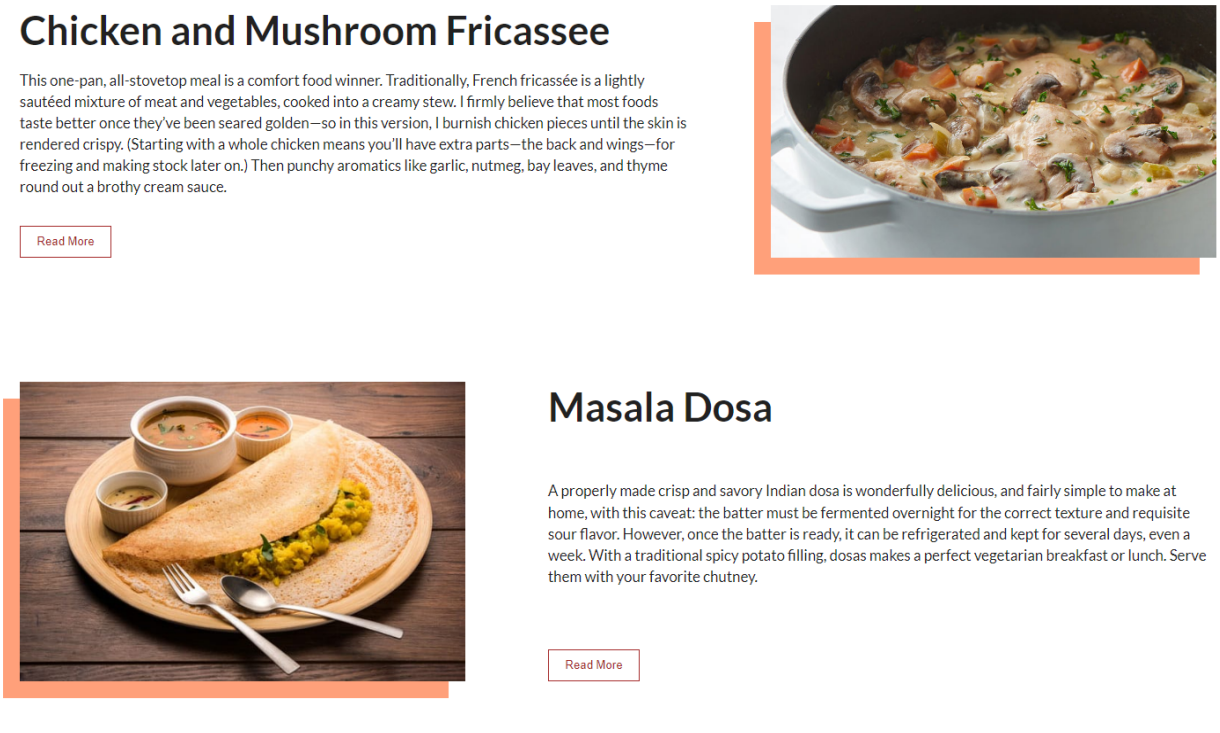


The logged-in user is displayed in the navigation bar with other navigation sections like About Us, Contact Us, and Categories Dropdown section. The user can view the active blogs on the home page with titles, descriptions, and images. All the blog details are fetched from the database with the “Posts” table. The “Posts” table includes columns like id, title, description, image, date, user id, and category as below.

Here, if you can check the image source link fetched from S3 is stored in the img column in RDS database.

id	title	descr	img	date	uid	cat
4	My Favorite Lemon Chicken Rec...	<p>As someone who has always been a f...	https://application-uploads.s3.amazonaws.com/5bddac385...	2023-04-28 01:40:19	2	american
5	Delicious Dumplings	<p>Recently, I had the pleasure of tryin...	https://application-uploads.s3.amazonaws.com/26f4610fb2...	2023-04-28 01:51:19	2	chinese
9	Chicken and Mushroom Fricassee	<p><span style="color: rgb(68, 68, 68);...	https://application-uploads.s3.amazonaws.com/a493bff6aa...	2023-04-28 17:19:01	6	french
10	Masala Dosa	<p><span style="color: rgb(34, 34, 34);...	https://application-uploads.s3.amazonaws.com/e807e3d81...	2023-04-28 17:21:36	6	indian
11	Chicken Pizza	<p><span style="color: rgba(0, 0, 0, 0.8...	https://application-uploads.s3.amazonaws.com/e89f4c86d0...	2023-04-28 17:31:24	6	italian

4.1.4 Food Blogs Page



The Food Blogs section includes all the active user's blog details with the title, description, and blog image. Here, every user can view all blogs and check details by clicking on "Read More" in individual blogs.

4.1.5 Individual User Blogs Page

Here there are two scenarios:

Scenario 1: When the blog owner wishes to edit/ delete blog details. The user who created the blog, can edit and delete his/ her own blog. There are Edit or Delete icons designed aside from the name of the blog. The respective blog author can only edit or delete their respective blog from our application. They can only view others' blog details but are restricted in managing blog details.

Other posts you may like

Masala Dosa

Read More

sagarp123
Posted 7 days ago

Masala Dosa

A properly made crisp and savory Indian dosa is wonderfully delicious, and fairly simple to make at home, with this caveat: the batter must be fermented overnight for the correct texture and requisite sour flavor. However, once the batter is ready, it can be refrigerated and kept for several days, even a week. With a traditional spicy potato filling, dosas makes a perfect vegetarian breakfast or lunch. Serve them with your favorite chutney.

Scenario 2: Different users cannot edit/ delete blogs of other users. They can only read other users blogs and have the only liability to edit/delete their own posts.

Other posts you may like

My Favorite Lemon Chicken Recipe!

Read More

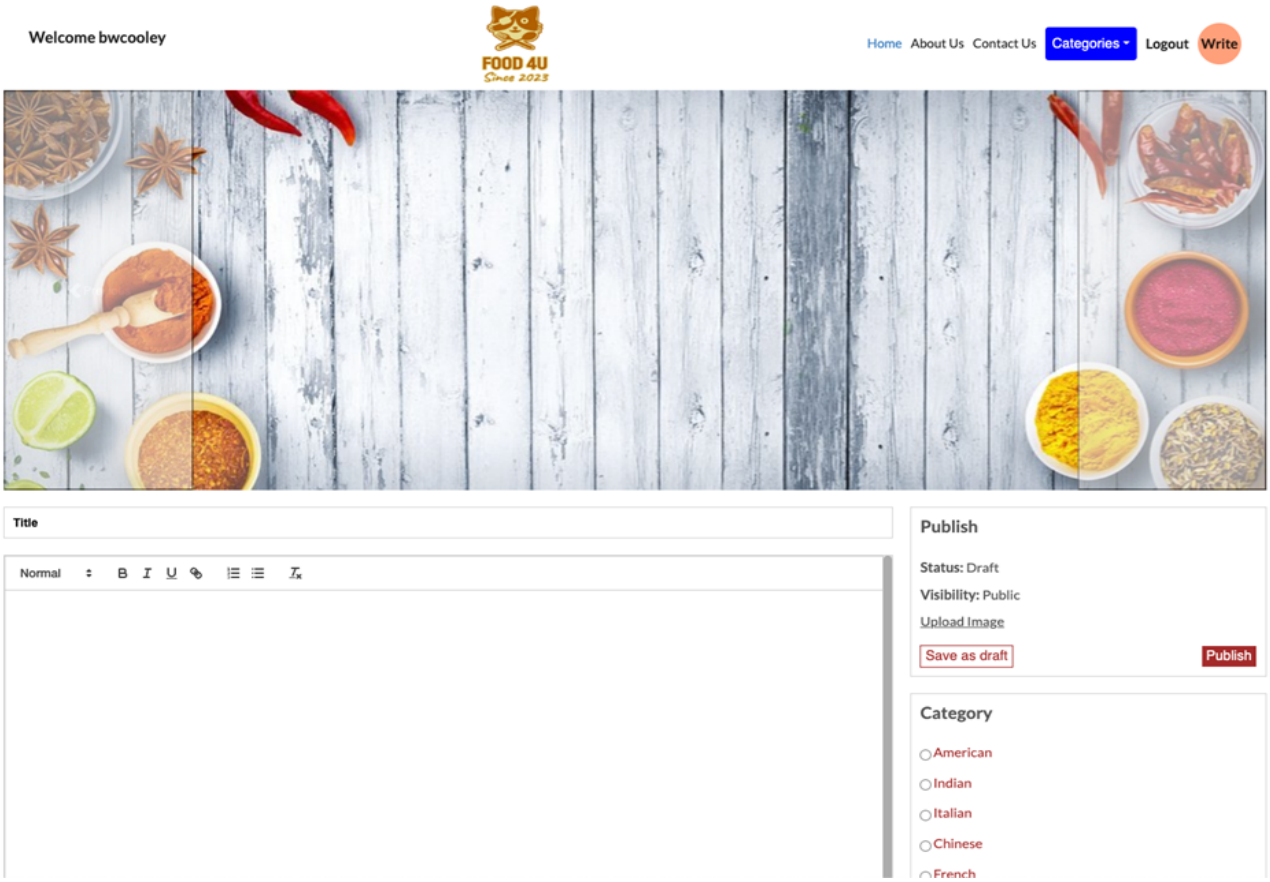
bcooley
Posted 19 hours ago

My Favorite Lemon Chicken Recipe!

As someone who has always been a fan of cooking and experimenting with new recipes, I am constantly on the lookout for new and exciting dishes to try out in my kitchen. Recently, I stumbled upon a recipe for lemon chicken that has quickly become one of my all-time favorites.There's just something about the combination of savory chicken with tangy, citrusy lemon that is so incredibly satisfying to the tastebuds. And this particular recipe takes that flavor profile to the next level with a few extra ingredients that really pack a punch.The first time I made this dish, I was blown away by how easy it was to put together. I had all of the ingredients on hand already, and it took less than 30 minutes to get everything prepped and cooking on the stovetop. And when it was all done, the aroma wafting through my kitchen was absolutely heavenly.When I finally took my first bite, I was hooked. The chicken was tender and juicy, with a crispy

Each user can view other remaining blogs from the same category from the section, “Other posts you may like”.

4.1.6 Create New Blog Page

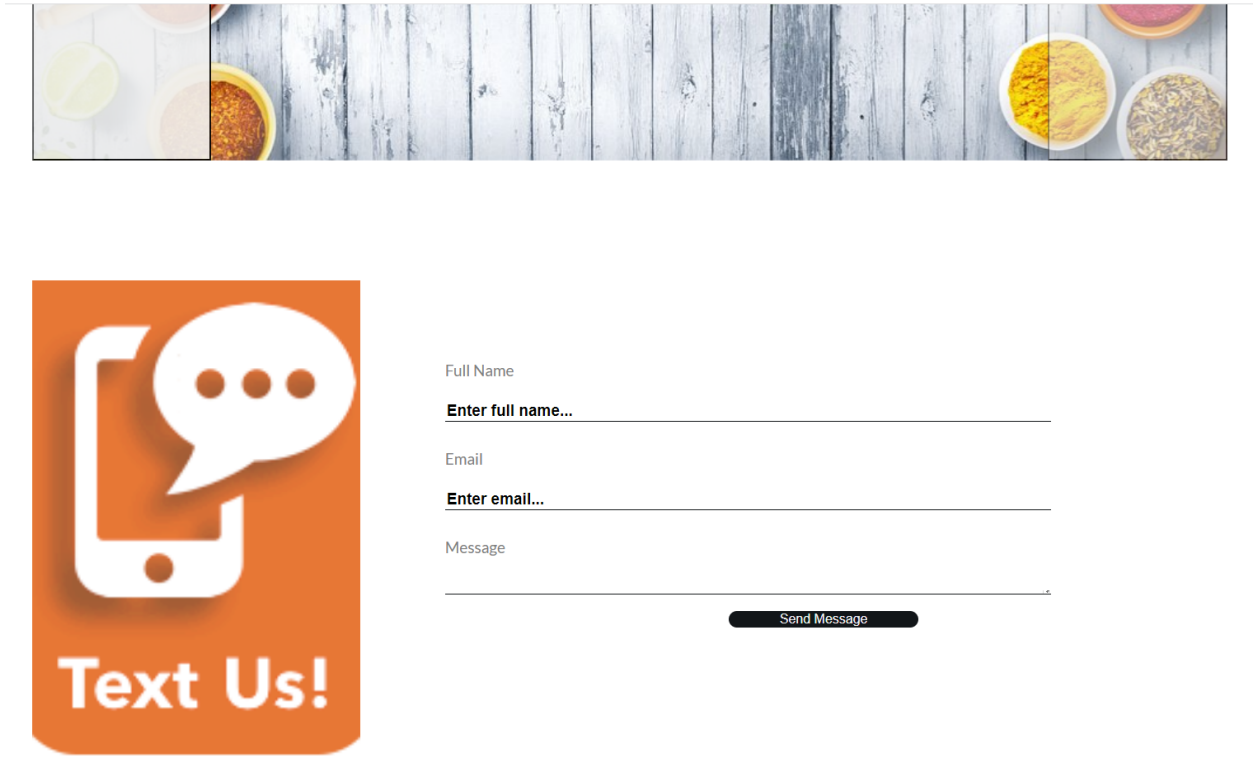


Each individual can create a new blog by adding the required details like the blog title, blog description, category, and blog image and publishing where all the details will be then added to the “posts” table. Users can also save blog details as a draft by clicking on “Save as a draft” where the person can edit details after coming back. This blog is then added to the Homepage and these details are marked as “Public” visibility which can be visible to all users but they are only allowed to edit or delete the blog details.

4.1.7 Edit Existing Blog Page

The author can edit the blog details by clicking the edit icon on the blog and modifying the details and again publishing the blog to the application.

4.1.8 Contact Us Page



CONTACT FOOD4U AT

OUR MAIN OFFICE
141 Broadway, North St,
Bloomington, IN 47404

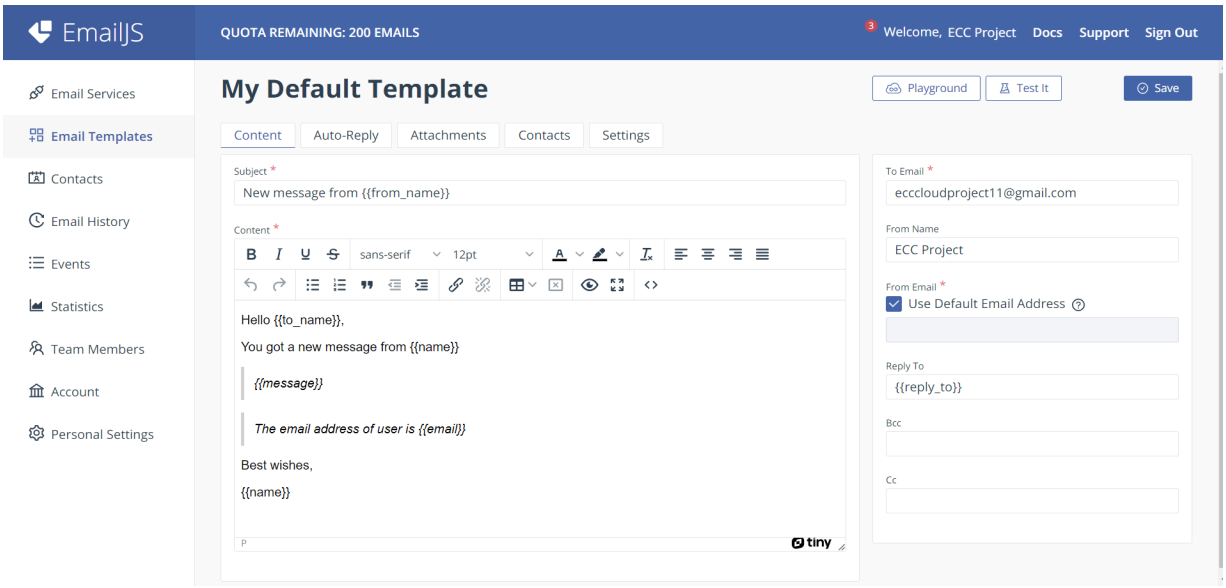
CONTACT PERSON
Sagar Prabhu (876-774-8432)

FAX
1-823-549-8843

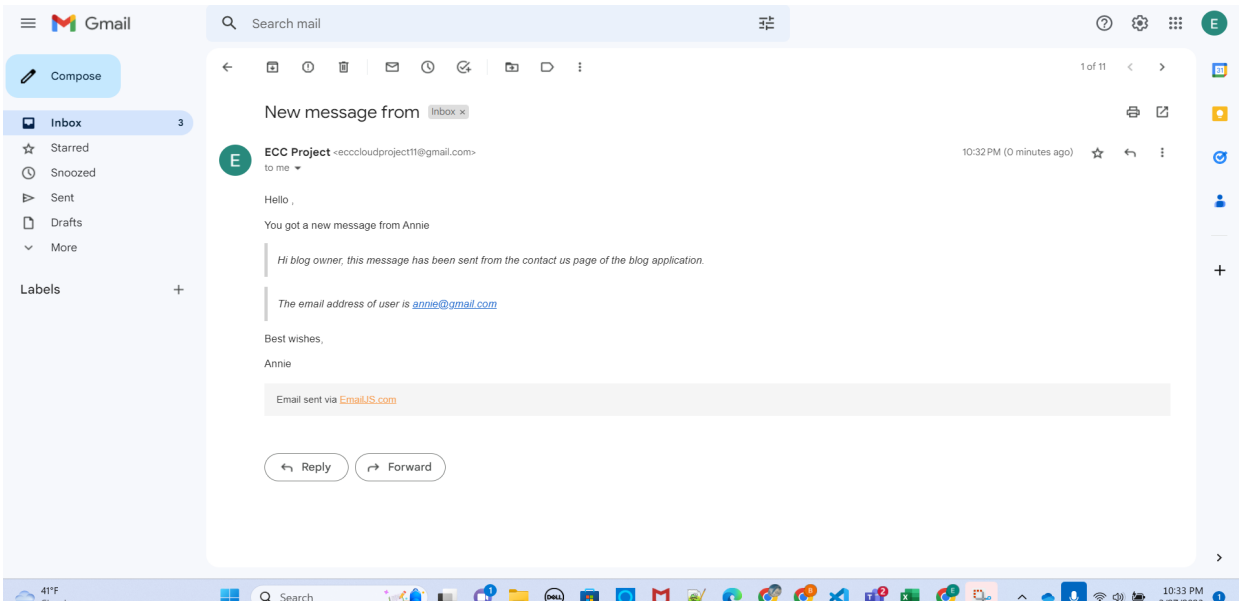
EMAIL
eccloudproject11@gmail.com

The contact us page enables the users to write a short message to the blog owner by entering their respective details like name, email address, and the message that is to be conveyed. Upon clicking on the ‘Send Message’ button, an email will be sent to the mailbox of the application owner.

In order to achieve this functionality, we have used the EmailJS API and set a template as below



An example of the email which is being sent to the blog owner is as below and this mail includes the name, email address, and the message which is sent by the user.



4.1.9 About US Page

This page gives the general information about our blog application with the story details and goals of developing the application with it's developers.

Our Story

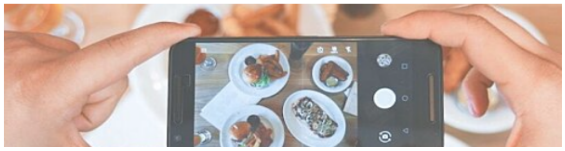
It's all started since 2012

Welcome to **Food 4U**, our delicious world of food blog!
From mouth-watering recipes to tantalizing food adventures,
our app is your ultimate guide to all things food.

Ready to indulge in a culinary journey that will tantalize your taste buds?
Look no further than our food blog app, where we serve up a smorgasbord
of delectable dishes and culinary creations that are sure to satisfy any appetite.

Our Goals

Our Food 4U blog displays delicious and remarkable food and beverages that meet the highest quality, freshness, and seasonality criteria while combining modern-creative and classic cooking traditions. By showcasing warmth, graciousness, efficiency, skill, professionalism, and integrity in our work, we will continually serve our consumers with exceptional service. To have every customer who comes through our website leave impressed and are excited to come back again.



Our Developers

Sagar Prabhu

Cloud & Application Developer

Application Architect

Brad Cooley

Cloud Architect & Developer

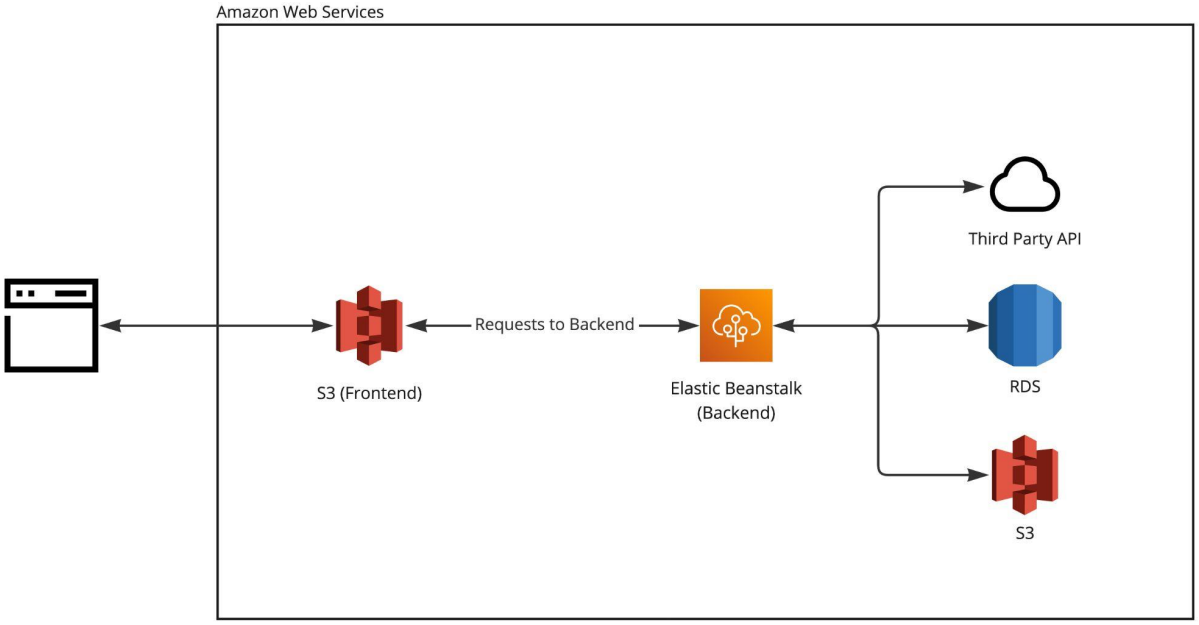
Bhakti Patrawala

Application Developer

4.2 Cloud Services

We were able to provision and automate the creation of our cloud services through infrastructure-as-code (IaC) as well as the AWS Command Line Interface (CLI). We detail these services below and what we did for their setup.

4.2.1 Architecture Changes



As we began the development of the application, we decided to avoid making the backend in a microservice architecture and instead opted for a monolithic architecture. We chose this because of two main factors, simplicity, and cost-effectiveness. The new architecture can be seen above with API Gateway and Lambda Functions no longer a part of the architecture.

4.2.1.1 Simplicity of a Monolithic Architecture

Going with our original microservice architecture meant we would have to be diligent upfront about how we wanted to design the data calls and data models. Because we didn't have a clear design in mind and figured that both of those things could change throughout the development process, we opted for a monolithic architecture. This gives us the benefit of simplicity throughout the development, deployment, and maintenance processes because we have consolidated the business logic and data storage into one place.

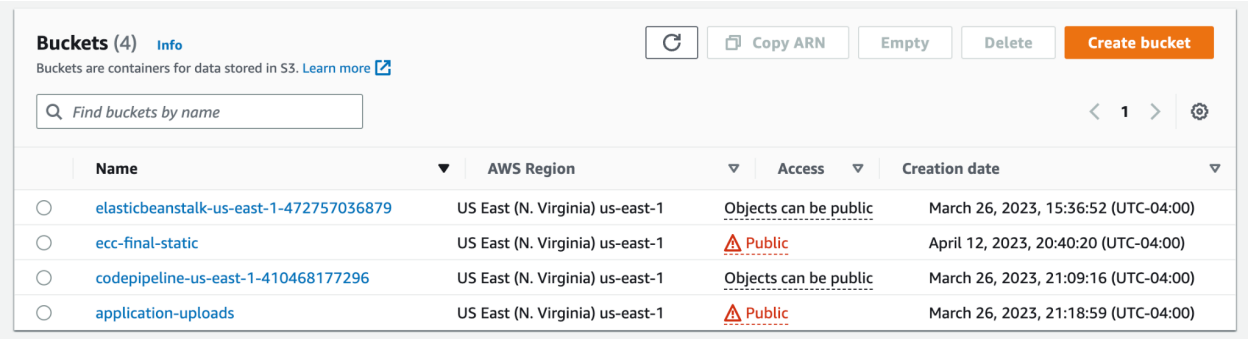
4.2.1.2 Lower Cost of a Monolithic Architecture

Microservice architectures can become expensive, both technically and financially, because of the number of resources that have to be provisioned and maintained. Moving to a monolithic architecture allowed us to eliminate two of our cloud services and keep the entire backend inside of Elastic Beanstalk. Moving to a single service managing the backend also means less cost in critical service monitoring utilizing something like CloudWatch or CloudAlerts.

4.2.2 Simple Storage Service (S3)

For the setup of Simple Storage Service (S3), we only had to create two buckets as the other two were provisioned from other services. The first bucket we created was to store any images that the user uploads to the application. This could include images for their blog post or a profile picture. The second bucket that was setup was for hosting the front end of the website. Since the website was hosted statically, we could easily just drop in the built files and mark the bucket as one that was statically hosting a website. This gave us a public facing URL and allowed for all traffic to reach every object inside the bucket.

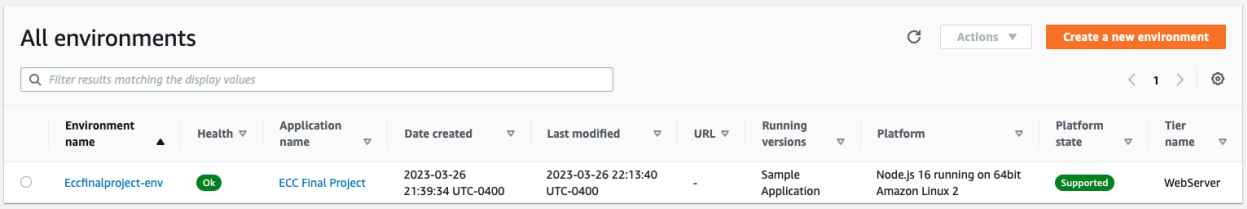
The other two buckets created for us are a result of other services, CodePipeline and Elastic Beanstalk. CodePipeline creates a bucket for the bundled and built code to reside as well as the clone that it makes of the repo when it first pulls it in from GitHub. Elastic Beanstalk takes the code from this bucket and then creates it's own bucket for health logs and other files related to the provisioning, running, and monitoring of the EC2 instances it's managing. Below is a screenshot of the four buckets in the US East 1 region inside the AWS console.



4.2.3 Elastic Beanstalk

Elastic Beanstalk was one of the easier cloud services to provision because of the AWS CLI and Elastic Beanstalk CLI (EB CLI). Due to free-tier limitations, we had to edit the service after it was provisioned so that there was no scaling and it was only utilizing t2.micro instances. It was setup to accept a bundled application that was stored in an S3 bucket; our code was placed into an S3 bucket from CodePipeline and was only the Express backend.

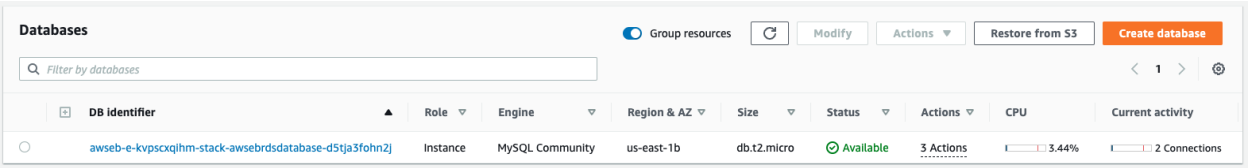
The publicly facing URL was actually hooked up to an Elastic Load Balancer (nginx) where, if we had more than a single instance, the traffic would be routed to a an under-utilized backend for the request to be processed. So, in our architecture, the frontend would send requests to the backend URL where the Elastic Load Balancer would then route traffic to a specific EC2 instance for the request to be processed and a response returned. Below is a screenshot of our *ECC Final Project* application and associated environment.



4.2.4 Relational Database Service (RDS)

Our Relation Database Service (RDS) instance was originally created and provisioned using infrastructure-as-code (IaC), but this way wouldn't allow us to do a free-tier eligible RDS instance. So, we had to delete that and instead let Elastic Beanstalk create and provision it. The Elastic Beanstalk setup, it allowed us to specify the compute we wanted RDS to run on along with the total size of space available for the MySQL instance.

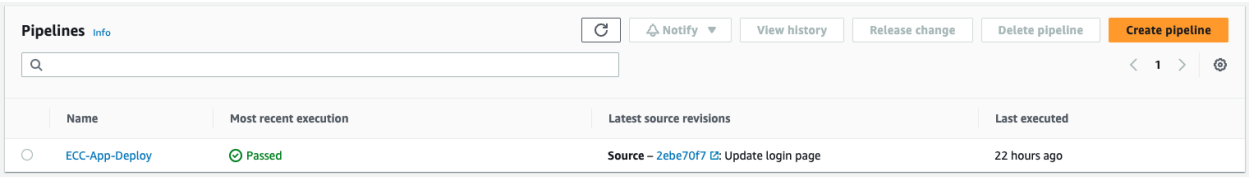
There were benefits beyond the free-tier aspect as Elastic Beanstalk also provisioned a custom security group so that the backend (all EC2 instances within the Elastic Beanstalk application) could talk to each other. Seeing that security groups and understanding how they work *well* can be a hard task on its own, having this done for us was quite nice. Below is a screenshot from the RDS dashboard of our database.



4.2.5 CodePipeline

This was a service that was not originally in our plan, but as we explored the CI/CD landscape, we decided to change our approach to prefer cloud-native solutions within the AWS ecosystem. The creation of the CodePipeline was one of the only things not done with IaC or the AWS CLI but instead was done through the AWS Web Console. We did this because it was easier to authenticate with GitHub than dealing with permission keys and access tokens.

The purpose of CodePipeline was to automate our CI/CD process, and specifically the later part of that (the deployment). Setting up this service allowed for our Elastic Beanstalk to just listen to an S3 bucket for new versions and then it would deploy those to the instances. Originally, we looked at using GitHub Actions, but dealing with permission keys, access codes, and secret access codes was not worth the hassle to have CI/CD for this project. However, upon looking into Amazon's offering, it seemed like the best choice for us along with using a service that didn't need to take up a bunch of development time to get working. Below is a screenshot of our repository currently passing build steps and deploying to Elastic Beanstalk (by way of an S3 bucket)



4.3 Code Hosting and Deployment

As discussed in section 4.2.5, originally, our architecture called for creating CI/CD pipelines using GitHub Actions. We pivoted to AWS CodePipelines for the CI/CD processes related to our application. However, this transition happened after the initial setup of the code repository. Instead of redoing the setup and configuration of the repository to move it from github.com to IU's Enterprise GitHub, we decided to mirror the repository into IU GitHub and continue development and deployment on github.com. Keeping the code in a github.com repository allows us more freedom and control over the processes, applications, and access to the code. The enterprise version that IU has limited us in some areas where we need flexibility. The entire aspect of GitHub Actions not being able to be used because we would have to supply our own runners/compute power is a prime example of this.

4.3.1 Understanding Cloud Deployment Procedures

Originally, we had planned on using GitHub Actions for all of our build procedures and our general CI/CD for the web application. However, throughout the setup process, it made more sense to utilize a cloud-native solution like CodePipelines from AWS. The main purpose of this was the simplicity of authentication tokens, and the runners available to us for free within AWS were better and more customizable than the ones available through GitHub Actions.

4.3.1.1 GitHub Actions

GitHub Actions provide an easy and simple way to stand up CI/CD pipelines for a code repository. They are free (up to a certain level) for general GitHub users. Initially, we chose to use GH Actions because of their simplicity and ability for us to iterate fast. Team members also had experience with them and knew how to create workflows using them. However, the deployment process was a little more complicated than we initially thought due to our overall application design and choosing to go with a monolithic backend. This meant a lot of the standard and lightweight GH Actions were not suitable for our use case. In addition to the complexity that the monolithic architecture brought in terms of the CI/CD process, GitHub Acces Tokens into AWS were fairly difficult because of AWS IAM roles and permissions. Because of this, we decided to transition to a cloud-native solution inside of AWS, CodePipelines.

4.3.1.2 CodePipelines

AWS CodePipelines is a cloud-native solution for CI/CD processes. We chose to pivot to this technology because of the robust feature set, tight security controls, and flexibility it gave us to customize our build processes. The biggest benefit of CodePipelines is being able to choose between AWS's native build solution CodeBuild or a more industry standard in Jenkins. Having familiarity with Jenkins, we created a build pipeline that handles the nuances of a monolithic architecture along with building and deploying a React app with lots of dependencies. We can make this free tier eligible by allowing Jenkins to only run on t2.micro compute, which is plenty for our building processes.

5. Future Work

In addition to the existing features, there are various other functionalities that can be implemented to make the blog application more lucrative from a business perspective. The following additions can be made to application in the future.

1. Adding more features and functionalities to the blog application, such as social media sharing, comments and search functionality.
2 Putting data analytics and visualization technologies to use can help you understand user engagement and behavior patterns, which will help you enhance your content and marketing initiatives.
3. Improving the user interface and user experience of the application by incorporating user feedback and conducting usability testing.
4. A disaster recovery strategy must be in place to guarantee minimal downtime in the event of a calamity. Disaster recovery implementation using AWS can assist reduce downtime and guarantee that the data is always accessible.
5. The application can assist provide high availability and lower latency by being deployed in several regions. To deploy the application in many regions and route traffic based on user location, utilize Amazon Route 53.
6. By employing more Amazon Web Services, integrating load balancing, and auto-scaling techniques, the application can be scaled to manage high volumes of traffic and data.

6. References

- https://aws.amazon.com/elasticbeanstalk/
- https://aws.amazon.com/getting-started/hands-on/create-mysql-db/
- https://flaviocopes.com/node-aws-s3-upload-image/
- https://stackabuse.com/building-a-rest-api-with-node-and-express/
- https://react.dev/blog/2023/03/16/introducing-react-dev
- https://www.bezkoder.com/react-node-express-mysql/
- https://medium.com/@arijit_chowdhury/basic-crud-app-setup-with-react-node-js-express-mysql-5e097e1145ff
- https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.deploy-existing-version.html
- https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/deploy-a-react-based-single-page-application-to-amazon-s3-and-cloudfront.html
- https://tinabu.medium.com/how-to-deploy-your-node-js-server-using-mysql-database-on-aws-mac-e64aaa2c86c