Architectural Blueprint and Implementation of 'Socials': A Scalable Social Media Platform on AWS

Safdar Ahmed, Mehran Banka, Deepjyot Singh, Xu Zhou

***Abstract*—This report details the development of a social media platform, named "Socials," which is implemented using various Amazon Web Services (AWS) to demonstrate the practical application of cloud infrastructure in deploying scalable social media applications. The primary objective was to replicate essential features of Instagram, such as user profiles, image uploading, post feeds, and interactive elements like likes and comments.**

I. Motivation

"Socials" is not just another social media platform; it's a revolution in how students interact online. Developed by a group of NYU students who were frustrated by the limitations of existing social media in academic settings, "Socials" provides a tailored environment where students can discuss projects, form study groups, and support each other's academic and extracurricular pursuits—all within a network that understands the rhythm of student life.

How can technology bring NYU students closer together? With this in mind, we developed a platform that not only allows students to connect socially but also serves as a bridge between various academic departments and student organizations.

Whether it's showcasing creative work, exchanging ideas, or organizing campus events, "Socials" is the go-to platform for students to engage with their peers in a meaningful way.

II. Introduction

In the fast-paced realm of digital communication, social media platforms serve as essential conduits for interaction, content sharing, and information dissemination. College students, a significant and active user demographic, utilize these platforms extensively for social connectivity and personal expression. In response to their specific needs, our project, "Socials," was conceived to reimagine an Instagram-like platform tailored for this audience, leveraging the comprehensive capabilities of Amazon Web Services (AWS).

The initial architectural strategy for "Socials" involved adopting a microservices architecture to allow for the flexible development, deployment, and scaling of independent services. However, as the project developed and the complexity of managing multiple services at scale became more apparent, it necessitated a shift towards a more robust orchestration solution. This shift led to the adoption of Kubernetes (K8s), a sophisticated container orchestration system that significantly enhances the deployment, scaling, and operational management of application containers across host clusters. While the current design still employs a microservices architecture, it is now underpinned by Kubernetes, unlike the earlier user-managed setup, providing a more structured and efficient management of the microservices.

This shift was motivated by the dynamic and high-volume interaction patterns of college students, who demand a platform that is not only functionally rich but also consistently reliable and scalable. Kubernetes offers advanced features such as automatic binpacking, self-healing, auto-scaling, and service discovery, which are crucial for maintaining the performance and availability of Socials under varying loads.

Utilizing AWS services such as Amazon EKS (Elastic Kubernetes Service) further empowered our team to seamlessly integrate Kubernetes, facilitating the management of clusters and simplifying containerized application deployment. This strategic pivot to Kubernetes underscores our commitment to providing a seamless, efficient user experience for college students, ensuring that Socials can handle their intensive media sharing and social interactions.

This report will elaborate on the developmental journey of Socials, from its initial microservices framework to its current Kubernetes-based architecture. It will detail the architectural decisions, the integration of AWS services, and the performance benefits realized, aiming to establish Socials as a scalable, secure, and user-centric platform that resonates strongly with college students.

III. Architecture

The architecture includes Amazon EKS, AWS Lambda, API Gateway, Amazon DynamoDB, AWS Rekognition, AWS OpenSearch and Amazon S3. Here, we detail each component's role and interaction with other services.

Endpoints and Database Design:
Detailed information about the API endpoints and the database schema:

- **Endpoints:**

```
POST /login
POST /register
GET /user/{userId}
POST /post
POST /comment
PUT /like
GET /search/{query}
GET /profile/{userId}
POST /follow/{userId}
```

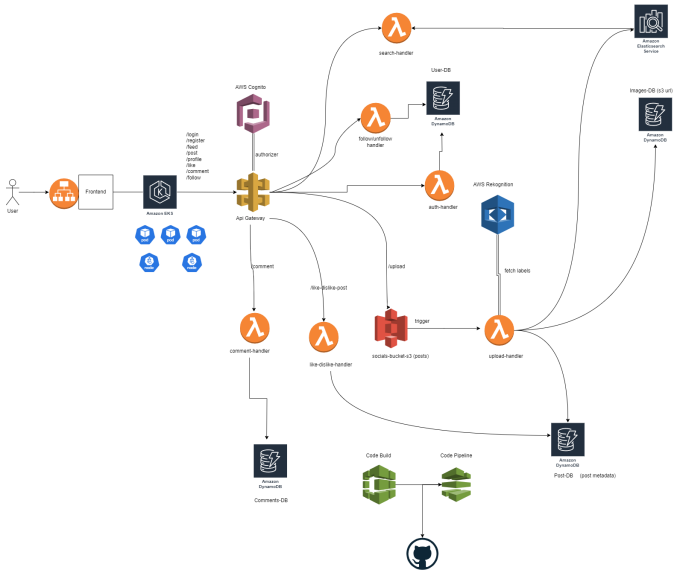

Fig. 1. Architecture Diagram

- **Database Schemas:**

1) User Database (User-DB):

```
user_id: String (Primary Key)
followers_ids: Set<String>
following_ids: Set<String>
post_ids: Set<String>
user_email: String
user_first_name: String
user_last_name: String
```

2) Post Database (Post-DB):

```
post_id: String (Primary Key)
caption: String
comment_ids: Set<String>
image_id: String
like_count: Integer
user_id: String
```

3) Image Data:

```
image_id: String (Primary Key)
s3_url: String
```

4) Comment Database (Comments-DB):

```
comment_id: String (Primary Key)
comment: String
comment_user_name: String
post_id: String
```

IV. Components

A. Amazon Elastic Kubernetes Service (EKS)

Amazon EKS is utilized to manage and orchestrate containerized applications, providing a highly scalable and fault-tolerant platform suitable for dynamic and complex applications. The use of Kubernetes via EKS enhances the deployment, scaling, and operations of application containers across clusters of hosts. Here are the specifics of the EKS deployment for "Socials":

- **Number of EKS Nodes:** The application employs 2 EKS nodes to distribute the workload and manage failover efficiently. This setup ensures that the system can handle traffic and process demands without any single point of failure.
- **Number of Pods:** A total of 3 pods are deployed to manage the frontend of the application. This allows for better load balancing and resource utilization, ensuring that the frontend remains responsive and reliable even under varying load conditions.
- **Machine Types for EKS Nodes:** The nodes are deployed on EC2 instances, specifically chosen for their balance of compute, memory, and network resources. While the specific instance types used can be tailored to the needs of the application, typical choices might include instances such as the t3.medium or m5.large, known for their performance and cost-efficiency.
- **Entry Point:** The entry point to "Socials" is managed by an Elastic Load Balancer (ELB), which effectively distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses. Managed within the EKS environment, the ELB ensures high availability and fault tolerance, automatically adjusting to the varying load and operational demands of the application.

While the current design still employs a microservices architecture, it is now underpinned by Kubernetes, which provides a more structured and efficient management of the microservices, particularly with the frontend services hosted on EKS.

B. AWS Lambda Functions

AWS Lambda offers a serverless execution model, removing the need for server provisioning and management while ensuring high availability and scalability. Lambda functions in this architecture are triggered by API Gateway requests, handling various backend tasks efficiently. Below is an expanded description of each function's role:

- **Authorizer:** This function manages user authentication and authorization. It verifies user credentials and tokens, ensuring that requests are made by authenticated and authorized users. This is crucial for maintaining the security and integrity of user data.
- **Auth-handler:** Handles additional security checks and complex authentication processes. It may be used for tasks like token validation, session management, and implementing multi-factor authentication, enhancing the security framework beyond basic login procedures.
- **Comment-handler:** Responsible for processing user comments on posts. This includes operations such as creating new comments, retrieving comments for a post, and managing updates or deletions of existing comments. Ensures that comment interactions are handled swiftly and efficiently.

- **Like-dislike-handler:** Manages user interactions related to liking and disliking posts. It updates the like and dislike counts on posts and ensures these metrics are accurately reflected in real-time on the user interface.
- **Upload-handler:** Manages the uploading of images and posts. This function triggers AWS Rekognition for image analysis to ensure content suitability and applies image tags for better categorization and retrieval. It also handles the storage of image metadata and the association of images with posts in DynamoDB.

These functions are designed to execute in response to specific events, making the backend highly responsive and capable of scaling automatically with the application's usage patterns. Each function is isolated to a single responsibility, promoting a clean and maintainable codebase.

C. AWS API Gateway

AWS API Gateway serves as a robust, fully managed gateway that acts as the single point of entry for all frontend requests. It's not only a critical component for routing requests to the appropriate AWS Lambda functions but also offers several key functionalities that enhance security, monitoring, and management of API traffic. Here are its expanded roles and features:

- **Request Routing:** API Gateway efficiently routes requests to the correct Lambda function based on the API endpoint accessed. This routing capability is central to managing the different aspects of the application such as user authentication, post interactions, and content management.
- **Security:** It integrates seamlessly with AWS Identity and Access Management (IAM) and AWS Cognito, providing robust security measures such as authentication and authorization checks before allowing access to backend resources. This ensures that all interactions are secure and that only legitimate users can access sensitive functions.
- **Rate Limiting:** API Gateway can enforce rate limiting to prevent abuse and to mitigate against potential DDoS attacks. This is essential for maintaining the application's availability and performance, especially under high traffic conditions.
- **Caching:** To improve the response time of requests, API Gateway provides caching capabilities. This reduces the number of calls made to backend services, thereby enhancing the overall efficiency and speed of the application.
- **Monitoring and Logging:** Integration with AWS CloudWatch allows for detailed monitoring and logging of all requests and responses. This visibility is crucial for debugging issues, understanding application usage patterns, and ensuring compliance with regulatory standards.
- **Version Management:** API Gateway supports multiple API versions and stages, allowing developers to test new versions of their APIs without affecting the current production environment. This is vital for continuous integration and delivery practices.

The use of AWS API Gateway thus ensures that the application's backend is not only scalable and secure but also highly efficient and reliable. Its comprehensive feature set supports a seamless user experience by effectively managing traffic, enhancing security, and providing necessary operational insights.

D. Amazon DynamoDB

Amazon DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale. It's a fully managed, serverless, and highly durable database service that fits perfectly with the needs of a dynamic social media application. Here are several key aspects of how DynamoDB is utilized in this architecture:

- **Performance and Scalability:** DynamoDB supports the needs of high-traffic applications by providing consistent, low-latency data access. This makes it ideal for the user, posts, and comments data, which require rapid read and write access due to the interactive nature of social media. Its ability to automatically scale up and down according to the application's demand ensures that the database performance remains optimal without manual intervention.
- **Fully Managed:** As a fully managed service, DynamoDB eliminates the administrative burdens of operating and scaling a distributed database. There are no servers to manage or software to install and maintain, which allows developers to focus more on application development rather than on database management.
- **Data Model Flexibility:** DynamoDB supports both key-value and document data structures, allowing it to store complex hierarchical data. This flexibility is particularly useful in social media settings, where the data model may evolve as new features are developed.
- **Built-in Security:** Security in DynamoDB is robust, featuring encryption at rest which secures your data from unauthorized access. Additionally, fine-grained access control can be managed using AWS Identity and Access Management (IAM), allowing precise control over who can access the database and how they can interact with it.
- **Integration with AWS Ecosystem:** DynamoDB integrates seamlessly with other AWS services used in the application, such as AWS Lambda for triggering data processing workflows, and Amazon API Gateway for data retrieval and updates. This integration facilitates a cohesive and efficient infrastructure.
- **Durability and Availability:** DynamoDB offers built-in fault tolerance as data is automatically replicated across multiple zones within a region. This ensures high availability and durability of the data, which is crucial for maintaining a reliable service in a social media application where data loss can significantly impact user experience.

E. Amazon S3

Amazon Simple Storage Service (S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. In the context of this social media application, S3 is primarily used for storing images, which are a central part of user posts and interactions. Here's a detailed look at how S3 is integrated and utilized:

- **Image Storage:** S3 provides a highly durable and available storage solution for images uploaded by users. Each image uploaded to a post is stored in S3, which guarantees durability, ensuring that the data is maintained with virtually no risk of losses.
- **Event-Driven Architecture:** The upload of images to S3 buckets triggers other processes within the architecture. For instance, the S3 bucket is configured to send notifications to AWS Lambda whenever new images are uploaded. This triggers the **upload-handler** Lambda function, which processes the images further, such as integrating with AWS Rekognition for content moderation and tagging.
- **Scalability and Performance:** Amazon S3 handles a large volume of data and high rates of requests from users across the globe without performance degradation. This capability is crucial for social media applications where the number of user-generated uploads can be extremely high and unpredictable.

Utilizing Amazon S3 thus supports the application's requirements for a robust, secure, and scalable storage solution for user-generated content, particularly images. Its integration into the system's event-driven architecture enhances the application's responsiveness and capability to process and manage content effectively.

F. AWS Rekognition

AWS Rekognition plays as an image analytics that uses machine learning to identify objects, people, text, scenes, and activities, as well as detect any inappropriate content. In "Socials", Rekognition plays a critical role in enhancing user content management:

- **Content Moderation:** Rekognition automatically screens images uploaded by users to detect inappropriate or unsafe content, which helps maintain community standards and comply with regulatory requirements.
- **Image Tagging:** It also automatically tags uploaded images based on their content. This feature aids in organizing and retrieving images more efficiently, enhancing user experience by enabling feature-rich search capabilities.
- **Integration with Lambda:** The integration of Rekognition with AWS Lambda facilitates an automated workflow where image analysis results can trigger further actions, such as alerting moderators or tagging images for further review, thereby automating much of the moderation process.

G. AWS Cognito

AWS Cognito is a comprehensive user identity and access management service that provides authentication, authorization, and user management for web and mobile applications. The service is integral to managing user identities and supporting secure and scalable user access in the social media application. Here are the key functionalities provided by AWS Cognito:

- **User Registration and Authentication:** Cognito facilitates the registration of new users and handles user authentication during the login process. It supports various authentication methods, including social identity providers (like Google, Facebook), SAML-based identity providers, and username-password logins.
- **Session Management:** Once a user is authenticated, Cognito manages user sessions, providing tokens for accessing APIs securely. It ensures that tokens are refreshed and that user sessions are valid, thereby maintaining secure access throughout the user's interaction with the application.
- **Security Features:** Cognito provides built-in security features such as multi-factor authentication (MFA), encryption of user data at rest, and the ability to detect and respond to abnormal user behaviors that may indicate a security risk.
- **Scalability:** As the application grows, Cognito scales to handle millions of users efficiently. Its distributed nature ensures that the user management system remains responsive and robust under all traffic conditions.
- **Customizability:** Cognito allows customization of workflows related to user management, such as email and SMS verifications, welcoming emails, and custom authentication challenges, making it adaptable to specific needs of the social media platform.

By leveraging AWS Cognito, the application ensures a seamless and secure user experience, supporting robust user registration, authentication, and management functionalities that are essential for any modern social media platform.

H. CI/CD and Infrastructure Management

The "Socials" application incorporates advanced AWS services for continuous integration and delivery (CI/CD) and infrastructure management to streamline development processes and ensure robust deployment practices.

- **AWS CodeBuild** AWS CodeBuild is a fully managed build service that compiles source code, runs tests, and produces software packages that are ready to deploy. In the "Socials" application, CodeBuild automates the compilation of code changes and ensures that they pass all set tests before deployment, enhancing code quality and reducing potential deployment issues.
- **AWS CodePipeline** AWS CodePipeline automates the stages of the software release process, including build, test, and deploy. For "Socials", CodePipeline is configured to trigger builds in CodeBuild whenever changes

are committed to the repository. It manages the workflow that pushes these changes through various stages from testing environments to production, ensuring a consistent and reliable deployment process.
- **AWS CloudFormation** AWS CloudFormation provides a way to use programming languages or a simple text file to model and provision, in an automated and secure manner, all the resources needed for applications across all regions and accounts. This service is used in "Socials" to define and provision AWS infrastructure using a declarative template. CloudFormation ensures that all resources are deployed in a controlled and predictable manner, significantly simplifying the process of managing and updating resources as the application scales.

These CI/CD and infrastructure management tools collectively enhance the operational agility of "Socials", allowing for seamless updates and maintenance of the application, minimizing downtime, and improving the overall deployment cycle.

I. OpenSearch

OpenSearch is a community-driven, open source search and analytics suite derived from Elasticsearch and Kibana. In the "Socials" application, OpenSearch plays a crucial role in enhancing the search capabilities, enabling users to perform efficient and effective searches across a wide array of social media content. Here are the key benefits and functionalities provided by OpenSearch:

- **Full-Text Search:** OpenSearch offers powerful full-text search capabilities that allow users to query posts, comments, and user profiles using complex queries for a more refined search experience.
- **Scalability:** It scales dynamically with the growing data and user base of the application, ensuring that search performance remains optimal even under heavy loads.
- **Real-Time Analytics:** OpenSearch supports real-time analytics which is essential for generating insights from user interactions and content trends within the application.
- **Integration:** Seamless integration with AWS services, particularly with AWS Lambda for processing and indexing new content and with Amazon S3 where data is stored, ensures a cohesive and efficient infrastructure.

Utilizing OpenSearch within the "Socials" application infrastructure significantly enhances user experience by providing quick and relevant search results, supporting the application's need to handle large volumes of data effectively.

J. Frontend Development

The frontend of "Socials" is designed to offer an intuitive and responsive user interface, crucial for engaging and retaining users. The choice of technologies and frameworks for the frontend was driven by several factors:

- **React:** Chosen for its component-based architecture, React allows for the development of highly interactive and dynamic user interfaces. Its vast ecosystem and community support make it an ideal choice for modern web applications.
- **Performance Optimization:** Techniques such as lazy loading, code splitting, and efficient handling of API calls are employed to optimize the performance and speed of the application.

These technology choices ensure that the frontend of "Socials" is not only appealing and user-friendly but also robust and scalable, aligning with the overall performance and scalability goals of the application.

V. Future Works

As the "Socials" continues to evolve, several enhancements are planned to improve user experience, system performance, and overall functionality. Below are the key areas identified for future development:

- **Enhanced Session Management:** Implementing more robust session management to ensure secure and efficient user experiences. This may include advanced session handling features such as session expiration notifications and automatic session renewal.
- **Frontend Rendering Improvements:** Optimizing the rendering of frontend pages to enhance user interaction and visual appeal. This could involve adopting modern web technologies like React or Angular for more dynamic and responsive interfaces.
- **Performance Testing:** Conducting extensive performance testing, especially under high load conditions, to ensure the application remains stable and responsive. This includes stress testing and load testing to identify potential bottlenecks and scalability issues.
- **Advanced Content Moderation:** Enhancing the capabilities of AWS Rekognition or integrating additional AI-based solutions for more sophisticated content moderation. This could help in better handling of nuanced scenarios and improve safety and compliance.
- **Personalization Features:** Developing more personalized content delivery algorithms that cater to individual user preferences and behaviors, potentially increasing user engagement and satisfaction.
- **API Optimization:** Refining API endpoints to reduce latency and increase throughput, which could involve implementing more efficient data retrieval methods or caching strategies.
- **Data Analytics:** Incorporating more comprehensive data analytics to gain insights into user behavior and application performance. This could guide data-driven decisions for future features and optimizations.

These initiatives will help in maintaining the competitiveness of the application while ensuring it meets the evolving needs of its user base.

VI. Conclusions

The "Socials" application leverages a robust and scalable AWS architecture to deliver a high-performance, secure, and reliable social media platform. Through the use of services like

Amazon EKS, AWS Lambda, API Gateway, DynamoDB, S3, Rekognition, and Cognito, the application effectively manages a wide array of functionalities ranging from user management and content delivery to real-time interactions and content moderation.

Amazon EKS ensures the system's scalability and reliability, hosting both the frontend and backend services. AWS Lambda functions handle backend processes efficiently, enabling a serverless architecture that scales automatically with the application's demand. The API Gateway orchestrates seamless communication between the frontend and backend, enhancing security and managing API traffic effectively.

DynamoDB provides a fast and flexible database solution that scales effortlessly to accommodate the growth of user data and interactions. Amazon S3 offers reliable and secure storage for the vast amounts of user-generated content, particularly images, while AWS Rekognition supports advanced content moderation and image tagging to maintain community standards.

AWS Cognito delivers robust user authentication and session management, crucial for protecting user data and ensuring a secure user experience. The planned future enhancements, including improved session management, frontend rendering, performance testing, and more sophisticated content moderation, highlight a commitment to continuous improvement and innovation.

In conclusion, the "Socials" application stands as a testament to the power of modern cloud architectures in building scalable, secure, and user-centric online platforms. As it continues to evolve, "Socials" is well-positioned to adapt to changing technological landscapes and user expectations, ensuring a vibrant and engaging social media experience.

REFERENCES

[1] https://docs.aws.amazon.com/whitepapers/latest/cicd-for-5g-networks-on-aws/cicd-on-aws.html
[2] https://docs.aws.amazon.com/eks/latest/userguide/getting-started.html
[3] https://docs.aws.amazon.com/lambda/
[4] https://docs.aws.amazon.com/dynamodb/
[5] https://docs.aws.amazon.com/s3/
[6] https://docs.aws.amazon.com/cognito/
[7] https://docs.aws.amazon.com/apigateway/
[8] https://docs.aws.amazon.com/rekognition/
[9] https://docs.aws.amazon.com/opensearch-service/
[10] https://docs.docker.com/
[11] https://react.dev/