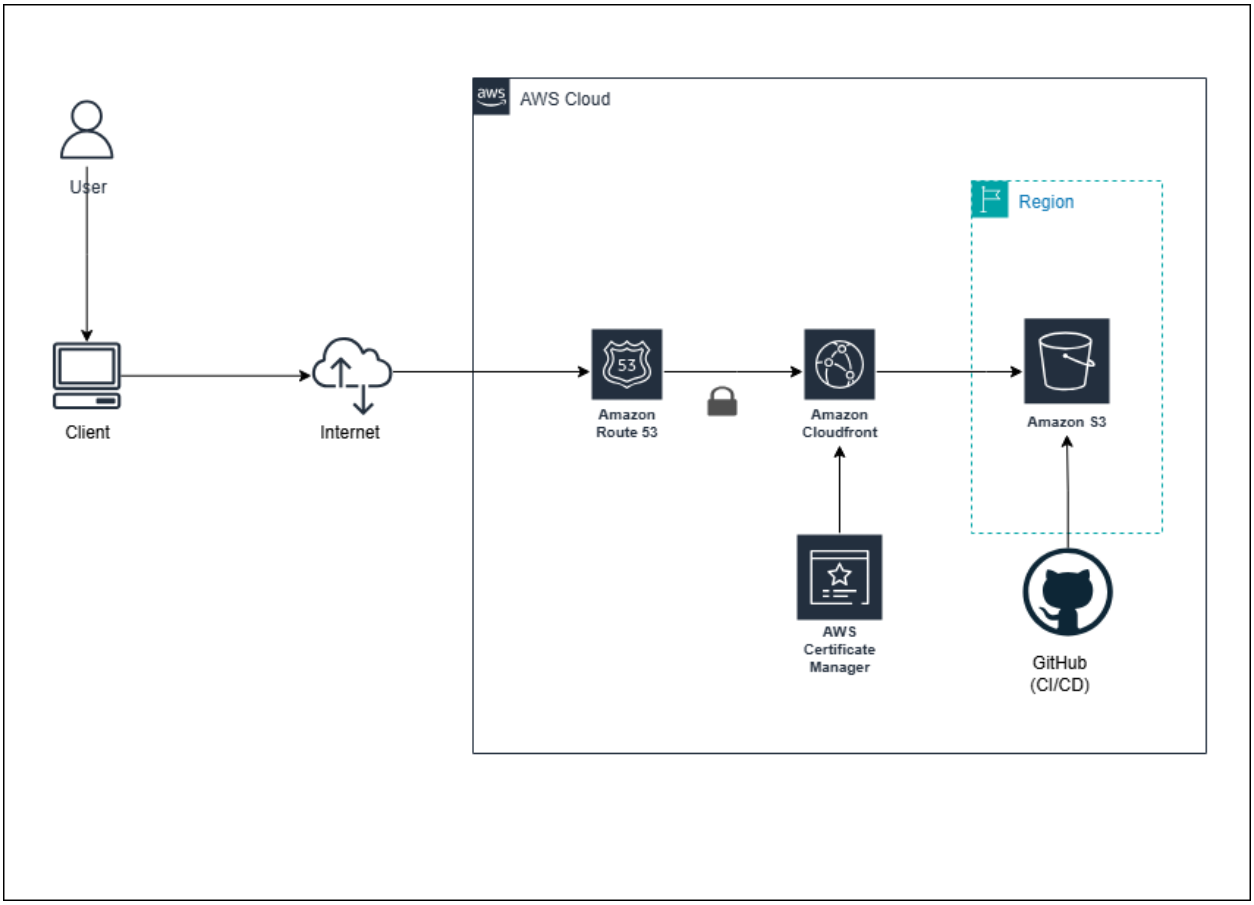
WEEK 1: DESIGNING A CLOUD ARCHITECTURE FOR A REACT FRONTEND APPLICATION

The proposed architecture implements a modern serverless approach for hosting static React applications, utilizing AWS global infrastructure for optimal performance and reliability.



EXPLANATION OF AWS SERVICES USED AND REASONS FOR SELECTION

1. Amazon S3 (Simple Storage Service): This is an object storage service offering industry-leading scalability, data availability, security, and performance. It is a static website hosting service that allows you to store and serve your React app files. Amazon S3 stores data as objects within buckets.

Reasons for selection:

i. Cost Efficiency: No server management costs, pay only for storage used.
ii. Scalability: Automatically scales to handle any amount of traffic.

iii. Integration: Native integration with CloudFront and other AWS services.
iv. Simplicity: Perfect match for static React applications after build process.

2. Amazon CloudFront: is a content delivery network (CDN) that accelerates delivery of static and dynamic web content to end users. CloudFront delivers content through a worldwide network of data centers called edge locations. It delivers the React application to users with low latency and high transfer speeds.

Reasons for selection:

i. Caching: Intelligent caching reduces S3 costs and improves user experience.
ii. SSL/TLS: Seamless integration with ACM for HTTPS termination.
iii. Performance: Reducing latency by up to 75%.
iv. Security: Built-in DDoS protection and Web Application Firewall integration.

3. Amazon Route 53: Amazon Route 53 is a managed Domain Name System (DNS) service. It is responsible for mapping the custom domain name to the CloudFront distribution. Route 53 was selected because it provides reliable, low-latency DNS resolution and integrates directly with other AWS services.

Reasons for selection:

i. Performance: DNS queries resolved in under 100ms globally.
ii. Reliability: 100% uptime SLA with anycast network.
iii. Health Checks: Monitor application health and route traffic accordingly.
iv. Advanced Routing: Support for weighted, latency-based, and geolocation routing.
v. Integration: Native AWS service integration.

4. AWS Certificate Manager (ACM): AWS Certificate Manager provisions and manages SSL/TLS certificates at no extra cost. It enables HTTPS for the React application when integrated with CloudFront, ensuring secure communication between users and the application. It automates certificate renewal, integrates seamlessly with CloudFront, and eliminates the complexity of manually managing certificates, ensuring both security and compliance.

i. Automation: Automatic certificate renewal (no manual intervention required)
ii. Security: Industry-standard encryption. It provides free SSL/TLS certificates.
iii. Integration: Seamless CloudFront integration

5. GitHub Actions (CI/CD Pipeline)**:** It allows developers to create workflows for Continuous Integration (CI) and Continuous Deployment (CD). In this architecture, GitHub Actions can be configured to automatically build the React application and deploy the generated static files to the Amazon S3 bucket whenever new code is pushed to the main branch of the repository.

Reasons for selection:

i. Automates the entire build and deployment process, reducing manual intervention.
ii. Ensures the application in S3 is always up to date with the latest code changes.
iii. Provides integration with AWS services.
iv. Allows workflow customization for testing, building, and invalidating CloudFront cache after each deployment.