Proposal to make the application available in an AWS environment.

The application of this repository must meet the requirements of high availability, low latency and the lowest possible cost
The proposal is that this application runs on a serverless architecture.
I will consider that the application runs globally, that is, in multiple regions.
Based on this assumption, the recommendation of the set of AWS services that meet these requirements are AWS Lambda and Amazon API Gateway, for the following reasons:

1 - High Availability: AWS Lambda is highly available and managed by AWS across multiple Availability Zones (AZs) within a region. This ensures that your functions are available in multiple locations to serve requests from users around the world.

2 - Low Latency: AWS Lambda is designed to execute functions close to users, reducing response latency. Additionally, the Amazon API Gateway, which can be integrated with Lambda, offers regional and edge-optimized API endpoints, allowing you to deploy functions closer to users regardless of their global location.

3 - Cost Optimization: The AWS Lambda pricing model, where you pay only for compute time used, remains advantageous for global applications. The automatic scaling also allows you to efficiently handle traffic spikes from different regions, avoiding unnecessary costs during periods of low demand.

4 - Simplified Management: AWS Lambda continues to offer the advantage of simplified management and reduced operational overhead, regardless of the global scale of the application. The absence of infrastructure management concerns facilitates maintenance and development across different regions.

5 - Stateless and Scalability: The stateless architecture of Lambda aligns with the scalable and global nature of modern applications. Each function invocation is independent, allowing them to be replicated and executed in any region quickly and efficiently.

The Amazon API Gateway plays a crucial role in serverless architecture when used together with AWS Lambda. It acts as an intermediary between clients (such as applications, browsers, or devices) and the AWS Lambda functions that execute the business logic.
By integrating AWS Lambda with the Amazon API Gateway, you can create custom API endpoints for your Lambda functions, allowing HTTP requests to be directed to the corresponding functions. This enables you to expose your application's functionality through a RESTful API, which is easier for clients to consume.
The Amazon API Gateway also provides powerful features, such as authentication and authorization, allowing you to control access to your APIs. It supports different types of authentication, such as API keys, access

tokens, or integration with identity providers, ensuring that only authorized users can access your Lambda functions.

Additionally, the API Gateway allows you to configure rate limiting policies and quotas to control incoming traffic to the APIs. This is especially important to ensure that your application is not overwhelmed with excessive requests.

Another advantage of the Amazon API Gateway is the ability to create edge-optimized APIs. These APIs are deployed to AWS's global Points of Presence, allowing client requests to be served with low latency, regardless of their location.