Challenge

By Jorge Vega

Solution and discussion of available options:

For this challenge, I chose to build a serverless solution consisting of an API Gateway integrated with a Lambda function written in TypeScript. This architecture allows the Lambda function to focus solely on its core functionality: receiving a dynamic string and returning an output string that incorporates the provided input. All aspects related to formatting the input and rendering the response in the browser are fully handled by API Gateway through the use of mapping templates.

Another possible solution using serverless:

Using Lambda function URLs instead of API Gateway would have allowed for a simpler and more cost-effective solution. However, I chose not to go with this approach for the following reasons:

1. I already have experience working with Lambda and API Gateway integrations, so this approach felt more natural and straightforward for me.
2. Combining Lambda with API Gateway represents a more realistic scenario, as it is the standard architecture for serverless applications on AWS.
3. Lambda function URLs would require handling request parsing and formatting directly within the Lambda function, reducing reusability. I prefer to keep each component focused on its specific responsibility.

Things I could add to embellish the solution:

1. Implement more specific validations for the dynamic string. Currently, if it's missing, the lambda simply display an empty string. I would prefer to provide a clear and descriptive error message about the string being required.
2. Add unit tests to validate that the CloudFormation template synthesized by CDK contains the expected resources, such as the Lambda function, the API Gateway, and the GET method, among others.
3. For more complex solutions, I would separate the resource definitions into different files to keep the main stack file simpler and more readable.