Challenge

By Jorge Vega

Solution and discussion of available options:

For this challenge, I chose to build a serverless solution using API Gateway integrated with a Lambda function written in TypeScript. The Lambda function retrieves a dynamic string from SSM Parameter Store, focusing solely on its core functionality: fetching the value and incorporating it into the response. All aspects of rendering the output in the browser are handled by API Gateway through mapping templates.

I decided to use SSM Parameter Store because this challenge requires only a single string, and SSM Parameter Store offers a free tier for up to 10,000 parameters per region, making it a simple and cost-effective choice.

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. 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.
2. For more complex solutions, I would separate the resource definitions into different files to keep the main stack file simpler and more readable.