Challenge questions:

Solution and discussion of available options

The solution I chose is of a serverless kind by using API-Gateway + Lambda + DynamoDB, meaning it operates with AWS resources that do not require an EC2/ECS instance to be running in order to host the web page, in other words this a zero server management solution that also scales automatically.

Other options were:

- Using S3 and Cloudfront: which provides static hosting, but it has the inconvenience that changing/updating the hosted page contents could require several object updates and triggering cache invalidation. Other moving parts could also be needed in order to make this option work in a fast and reliable way.
- Using EC2/ECS: which is too much of a heavyweight and expensive solution to host a simple HTTP page.

Reasons behind the decisions I made

- The API Gateway provides a fixed HTTP endpoint.
- Changes/updates made to a DynamoDB table are instantaneous, so they are picked up right away the next time the Lambda function runs.
- It’s simple and cost-efficient.
- Terraform easily deploys all required AWS resources/components.

How would I embellish the solution were I had more time

- I could add a POST /admin route with IAM auth (or Cognito) to update the string using the same API.
- I could use and configure CloudFront in front of the API for better global latency/performance by providing appropriate cache headers.
- I could set up CloudWatch observability tooling by creating dashboards and alarms.
- I could create a CI/CD pipeline to validate/format/plan/apply Terraform, and to lint the Lambda.
- I could also create an extra stage/step inside the pipeline to run Integration and/or Unit tests using Terraform’s Testing framework.
- I could add more parameterization to the written Terraform code (more variables) to provide more configuration flexibility to users using it.