✅ 1. Overview of the Solution

The goal was to provision infrastructure using Terraform on a cloud platform (AWS) to serve a publicly accessible HTML page with a dynamic string that can be updated without redeploying.

Example HTML served:

```
html
Copiar código
<h1>The saved string is Hello, Interviewer!</h
```

The infrastructure is fully serverless, cost-efficient, and easy to manage using Terraform.

☁️ 2. Architecture Overview

The solution uses:

- AWS Lambda – to generate and return the HTML
- API Gateway – to expose a public HTTPS endpoint
- SSM Parameter Store – to store the dynamic string
- Terraform – to fully define and provision the stack

📊 Architecture diagram:

```
text
Copiar código
User --> API Gateway --> Lambda --> SSM Parameter Store --> returns dynamic
HTML
```

💡 3. Options Considered

Several strategies were evaluated to satisfy the requirement of serving a modifiable HTML string:

A. Static HTML with S3 Bucket (📦 Rejected)

- Simple and cheap
- ❌ Requires redeployment or manual file update to change the string

B. Containerized Web App (ECS / EKS) (🐳 Rejected)

- Fully customizable
- ❌ Overkill for simple HTML string
- ❌ Involves managing networking, containers

✅ C. Lambda + API Gateway + SSM (Selected)

- Serverless and dynamic
- No unnecessary infrastructure
- String can be changed anytime via CLI or Console

🧠 4. Design Decisions

- Terraform was chosen for reproducibility and automation
- Lambda makes it easy to supply runtime logic without maintaining a server
- SSM Parameter Store enables updating the string without code changes
- AWS API Gateway provides HTTPS public access to trigger the Lambda

📌 The dynamic string is injected using environment variables to tell Lambda which parameter name to fetch.

🛠️ 5. How to Use the System

Deploy with:

```
bash
Copiar código
terraform init terraform apply
```

After deployment, you'll get a public URL:

```
php
Ejecutar código
Copiar código
https://<api-id>.execute-api.<region>.amazonaws.com/prod/html
```

To update the string without redeploying:

```
bash
Copiar código
aws ssm put-parameter \ --name /dynamic/html/string \ --value "New string value" \
--type String \ --overwrite
```

Refresh browser — done ✅

🚀 6. If I Had More Time, I Would...

💡 EMBELLISHMENTS:

- Add a simple frontend to modify the dynamic string through a web form (POST → SSM)
- Add authentication layer (Cognito or IAM) to protect the dynamic string from unauthorized changes
- Expose multiple dynamic parameters based on user, headers, or query strings
- Organize into modules using `terraform modules` for better reusability
- CI/CD Integration using GitHub Actions for automatic deploy/testing
- CloudFront CDN + Custom domain for better performance

📌 7. Conclusion

The deployed infrastructure is dynamic, scalable, secure, and lightweight. All requirements are fulfilled:

- ✅ Public URL serving HTML
- ✅ Configurable string stored outside the application (SSM)
- ✅ Updates require no redeploy
- ✅ Infrastructure as Code (Terraform)
- ✅ Cost-effective Serverless architecture