AWS Static Website Deployment (IaC + CI/CD)

Documentacion Técnica del Proyecto

Omar Garcia

https://d1emj2ptwgcjc0.cloudfront.net/

Tabla de Contenido

1. Introduccion

Este documento describe de forma detallada la arquitectura, decisiones técnicas y flujos de trabajo implementados en el proyecto de despliegue de un sitio web estático serverless.

El objetivo es que cualquier persona técnica pueda entender qué se construyó, cómo funciona cada componente y por qué se tomaron las decisiones de diseño que se tomaron. El proyecto implementa una infraestructura completa en AWS siguiendo las mejores prácticas de la industria: Infraestructura como Código (IaC), entrega de contenido global mediante CDN, pipelines CI/CD automatizados y políticas estrictas de seguridad.

2. Visión General de la Arquitectura

La arquitectura se divide en cuatro capas principales que trabajan en conjunto:

- **Capa IaC:** Terraform aprovisiona toda la infraestructura en AWS de forma declarativa y reproducible.
- **Capa CI/CD:** GitHub Actions automatiza el build, push y deploy en cada cambio de código.
- **Capa de Almacenamiento:** Amazon S3 actúa como el origen privado de los datos (archivos HTML, CSS, JS).
- **Capa de Entrega y Seguridad:** Amazon CloudFront sirve el contenido a nivel mundial mediante Edge Locations, forzando HTTPS y restringiendo el acceso directo al origen.

Flujo de extremo a extremo:

Plaintext

```
Developer (git push a main)
  |
  v
GitHub Actions
  |-- infra.yml  --> terraform apply --> AWS (S3, CloudFront, IAM, ACM)
  |-- deploy.yml --> aws s3 sync     --> S3 Bucket (Origen Privado)
                 --> aws cloudfront create-invalidation --> CloudFront Edge Cache
  |
Usuario --> HTTPS --> CloudFront (CDN) --> Origin Access Control (OAC) --> S3 Bucket Privado
```

3. Infraestructura como Código (Terraform)

Terraform define toda la infraestructura en archivos .tf versionados junto al código. Esto garantiza que la infraestructura sea reproducible, auditable y fácil de modificar. El estado se almacena en Amazon S3 para que tanto el equipo local como GitHub Actions lean el mismo estado.

3.1 Estructura de Modulos

Módulo	Recursos que crea	Dependencias
storage	S3 Bucket, S3 Public Access Block, S3 Bucket Policy	Ninguna
CDN	CloudFront Distribution, Origin Access Control (OAC), ACM Certificate	bucket_id y ARN de storage
security	IAM Policies, IAM Role/User para CI/CD	ARNs del S3 Bucket y CloudFront Distribution

3.2 Almacenamiento (Amazon S3)

- **Bucket Privado:** Se activa Block Public Access en su totalidad. No existe acceso público desde internet hacia el bucket.
- **Bucket Policy:** Se configura una política restrictiva que solo permite la acción s3:GetObject si la petición proviene explícitamente del ARN de la distribución de CloudFront (mediante la condición aws:SourceArn).

3.3 Red de Entrega de Contenido (CloudFront)

- **Edge Caching:** Reduce drásticamente la latencia global cacheando los activos estáticos en servidores cercanos al usuario final.
- **Origin Access Control (OAC):** Es el mecanismo moderno implementado para autenticar las peticiones desde CloudFront hacia S3, garantizando que el origen permanezca invisible para el internet público.
- **Viewer Protocol Policy:** Configurado estrictamente en redirect-to-https para asegurar el cifrado en tránsito.

4. Decisiones de Seguridad

El proyecto aplica el principio de mínimo privilegio en cada capa:

Práctica	Implementación	Beneficio
Origen Oculto	S3 Block Public Access + CloudFront OAC	Superficie de ataque cero en el backend de almacenamiento. Evita scraping y descargas directas.
Cifrado en tránsito	CloudFront con certificados AWS Certificate Manager (ACM)	Todo el tráfico es HTTPS.
IAM de mínimo privilegio	Política personalizada para GitHub Actions	Sin AdministratorAccess. El pipeline solo tiene permisos de s3:PutObject, s3:ListBucket y cloudfront:CreateInvalidation.
Secrets cifrados	GitHub Secrets + Terraform sensitive	Nunca aparecen en los logs.

5. Pipeline CI/CD (GitHub Actions)

El ciclo de vida del desarrollo está completamente automatizado mediante dos workflows independientes, cada uno con responsabilidad clara.

5.1 Workflow de Infraestructura (infra.yml)

Se dispara únicamente cuando hay cambios en la carpeta infra/:

Evento	Comportamiento	Propósito
Pull Request a main	terraform plan (solo lectura)	Revisar qué cambios habrá antes de aprobar el PR
Push a main	terraform plan + terraform apply	Desplegar los cambios de infraestructura automáticamente

5.2 Workflow de Aplicación (deploy.yml)

Se dispara en cada push a la rama principal (con cambios en el código web). Ejecuta los siguientes pasos:

1. **Checkout:** descarga el código del repositorio.
2. **Configure AWS credentials:** configura el acceso a AWS usando los secrets del repositorio.
3. **Sync a S3:** Sincroniza el código fuente compilado o modificado hacia el bucket S3 privado mediante el comando aws s3 sync.
4. **Invalidación de Caché:** Ejecuta aws cloudfront create-invalidation para forzar a los nodos Edge de CloudFront a purgar la caché antigua y servir la versión recién desplegada.

6. Flujo Completo End-to-End

1. El desarrollador hace git push a main con cambios en la aplicación estática.
2. GitHub Actions dispara el workflow de despliegue automáticamente.
3. Se autentica en AWS de manera segura.
4. La plataforma sincroniza eficientemente solo los archivos modificados hacia Amazon S3.
5. Se lanza una orden de invalidación a CloudFront.
6. El usuario visita el dominio y ve la versión actualizada en segundos, servida localmente desde su Edge Location más cercana.

7. Estructura del Proyecto

Ruta	Descripción
src/	Código fuente HTML/CSS/JS del portfolio
infra/main.tf	Orquestación de módulos (storage, cdn, security)
infra/backend.tf	Backend remoto S3 para el estado de Terraform
infra/modules/storage/	S3 Bucket, bloqueos y políticas IAM
infra/modules/cdn/	Distribución CloudFront, OAC y Certificados SSL
.github/workflows/deploy.yml	Pipeline de sincronización a S3 e invalidación CDN
.github/workflows/infra.yml	Pipeline de infraestructura (terraform apply)

8. Stack Tecnologico

Tecnología	Rol en el proyecto
Terraform >= 1.5.0	Aprovisionamiento de toda la infraestructura (IaC)
Amazon S3	Origen de los archivos estáticos y estado remoto de Terraform
Amazon CloudFront	CDN, Edge Caching, OAC y enrutamiento HTTPS
AWS IAM	Identidad y permisos granulares de integración continua
GitHub Actions	CI/CD automatizado (sync, invalidation)

9. Despliegue Manual

Si se requiere recrear la infraestructura desde cero sin GitHub Actions:

Bash

```
# 1. Inicializar Terraform y configurar backend S3 [cite: 144]
terraform init [cite: 145]

# 2. Generar el plan de ejecución [cite: 146]
terraform plan -out=plan.out [cite: 147]

# 3. Aplicar la infraestructura [cite: 148]
terraform apply plan.out [cite: 149]

# 4. Desplegar aplicación e invalidar caché
aws s3 sync ./src s3://<bucket> --delete
aws cloudfront create-invalidation --distribution-id <DISTRIBUTION_ID> --paths "/*"
```

10. Glosario

Término	Definición
IaC	Infrastructure as Code. Práctica de definir la infraestructura en archivos de código versionables.
Terraform	Herramienta de IaC que aprovisiona infraestructura en múltiples proveedores cloud de forma declarativa.
CDN	Content Delivery Network. Red global de servidores que aceleran la entrega de contenido acercándolo al usuario.
OAC	Origin Access Control. Arquitectura recomendada por AWS para proteger orígenes S3 cuando se utiliza CloudFront.
Invalidación	Proceso que purga los archivos en la caché de los nodos Edge, forzando la recarga desde el origen S3.