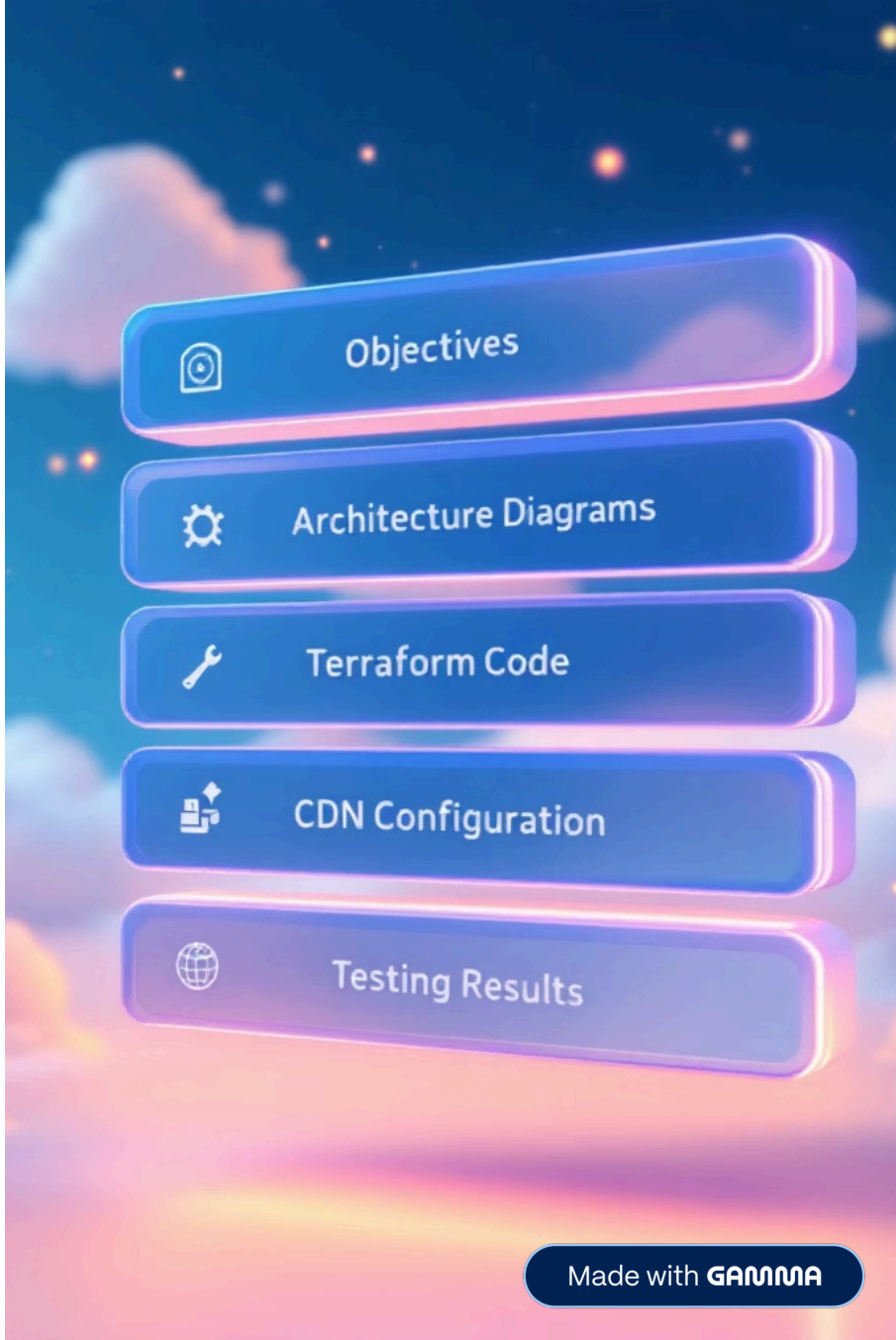
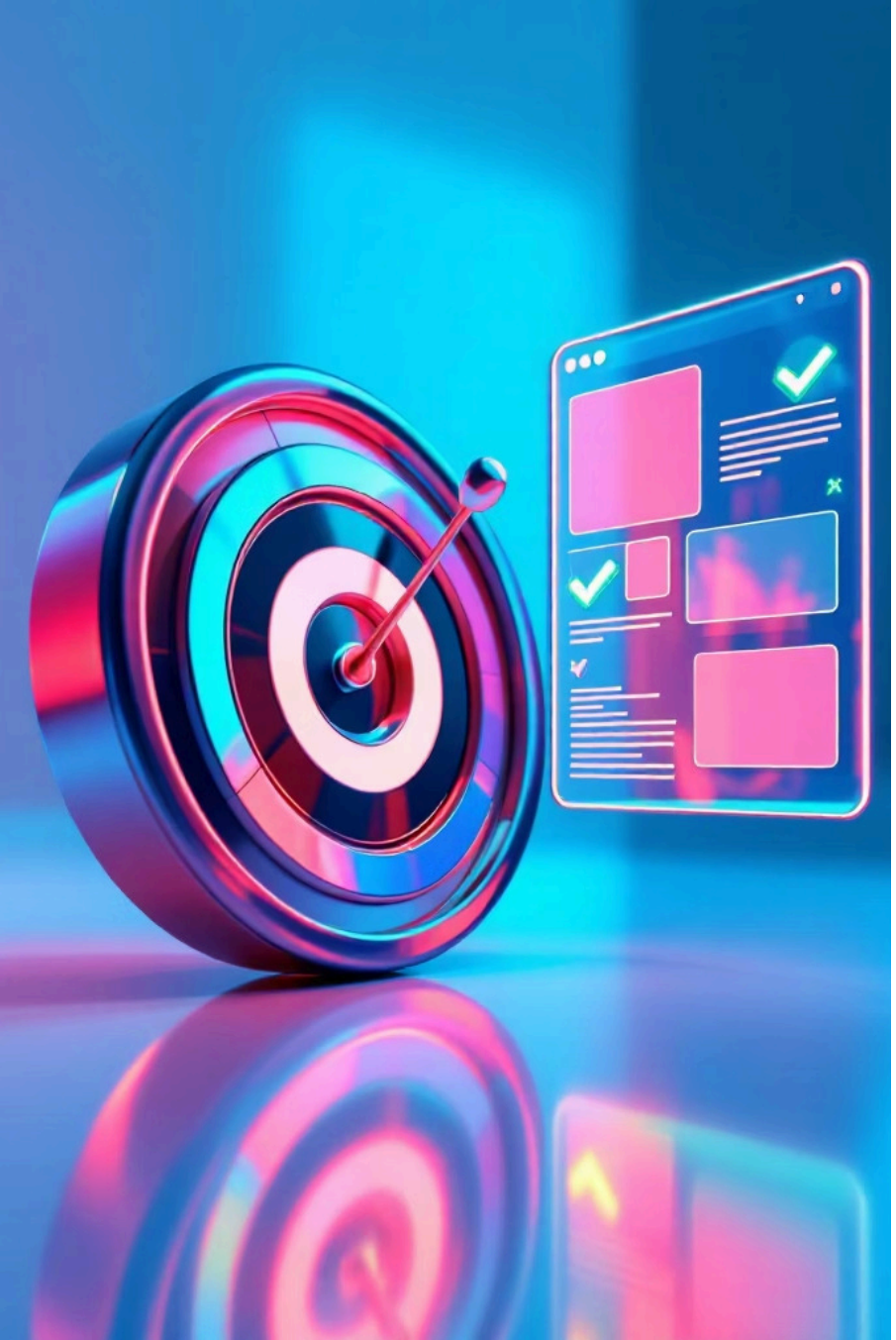
AI Website Cloud Deployment with AWS

Development & Implementation Phase — Fatima Ezzahrae Ezzouzi • 2026

Table of content





Objective

Host a static website on AWS via Amazon

Reliable object storage for site files and assets with durable hosting.

Use Terraform for automation

Infrastructure as Code to provision, version, and replicate the environment.

Global access & HTTPS via CloudFront

CDN-backed delivery with TLS termination and automatic HTTP→HTTPS redirects.

Tools & Services

Key components used for the deployment:

Service / Tool	Purpose
AWS S3	Store website files & host static website — origin for CloudFront
AWS CloudFront	Global CDN, TLS termination, edge caching & redirects
Terraform	Infrastructure as Code: create S3, CloudFront, policies, and uploads

References: https://aws.amazon.com/s3/ • https://aws.amazon.com/cloudfront/ • https://www.terraform.io/

Cloud Architecture

User requests are routed to CloudFront edge locations (TLS terminated at edge). CloudFront fetches static content from the S3 website origin when not cached, serving the site globally with low latency.

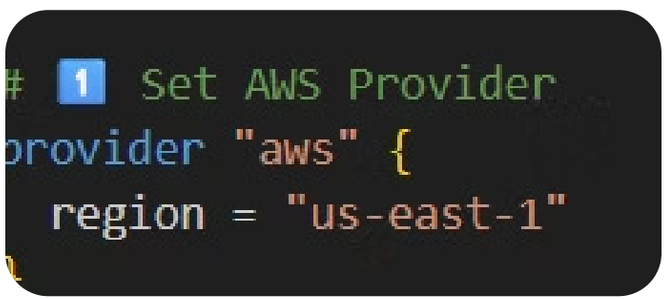
Terraform Implementation

What main.tf provisions

- S3 bucket for website files and hosting configuration
- CloudFront distribution with origin pointing to S3
- Bucket policy and OAI/Origin access configuration or public policy
- Optional automation to upload site assets

Notes

main.tf organizes resources, variables, and outputs. Use terraform init/plan/apply for repeatable deployments.
Placeholder: main.tf



AWS PROVIDER

Set up AWS credentials :

- **AWS Access Key**: Identify an account
- **AWS Secret Key**: The corresponding secret key pairs with an access key

Set an AWS access key and secret key as an environmental variable AWS_ACCESS_KEY_ID: **YOUR_ACCESS_KEY**

AWS_SECRET_ACCESS_KEY :"**YOUR_SECRET_KEY**"

S3 Bucket Setup

```hcl
# 2  Create S3 Bucket for Static Website
resource "aws_s3_bucket" "website" {
  bucket = "fatima-ai-website-2026"
  force_destroy = true
}

# 3 Disable S3 Block Public Access for this bucket
resource "aws_s3_bucket_public_access_block" "public_access" {
  bucket = aws_s3_bucket.website.id
  block_public_acls       = false
  block_public_policy     = false
  ignore_public_acls      = false
  restrict_public_buckets = false
}

# 4 Set S3 Bucket Policy — allow public read
resource "aws_s3_bucket_policy" "public_read" {
  bucket = aws_s3_bucket.website.id

  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [{
      Effect    = "Allow"
      Principal = "*"
      Action    = ["s3:GetObject"]
      Resource  = "${aws_s3_bucket.website.arn}/*"
    }]
  })
}

# 5 Configure S3 as a Static Website
resource "aws_s3_bucket_website_configuration" "website_config" {
  bucket = aws_s3_bucket.website.id
  index_document {
    suffix = "index.html"
  }
}
```

Created an S3 Bucket: fatima-ai-website-2026

- **Access**

  Bucket policy set for public read or served via CloudFront origin access pattern.

- **Static website hosting**

  Index document: index.html — configured in bucket properties.

```hcl
# 6 Upload index.html to S3
resource "aws_s3_object" "index" {
  bucket       = aws_s3_bucket.website.id
  key          = "index.html"
  source       = "index.html"
  content_type = "text/html"
}
```

- **Files uploaded**

  index.html pushed via Terraform

CloudFront Setup

```
resource "aws_cloudfront_distribution" "website_cdn" {
  origin {
    domain_name = aws_s3_bucket.website.website_endpoint
    origin_id   = "s3-website-fatima-ai"

    custom_origin_config {
      http_port              = 80
      https_port             = 443
      origin_protocol_policy = "http-only"
      origin_ssl_protocols   = ["TLSv1.2"]
    }
  }

  enabled             = true
  is_ipv6_enabled     = true
  default_root_object = "index.html"
```

01

Create distribution

Origin: S3 website endpoint or origin access identity. Default root object: index.html.

```
  default_cache_behavior {
    allowed_methods  = ["GET", "HEAD"]
    cached_methods   = ["GET", "HEAD"]
    target_origin_id = "s3-website-fatima-ai"

    viewer_protocol_policy = "redirect-to-https"

    forwarded_values {
      query_string = false
      cookies {
        forward = "none"
      }
    }
  }
viewer_certificate {
  cloudfront_default_certificate = true  # Use default
}
  restrictions {
    geo_restriction {
      restriction_type = "none"
    }
  }

  tags = {
    Name = "AI-Website-CDN"
  }
}
```

01

Enable HTTPS

Use default CloudFront certificate (*.cloudfront.net) or ACM for custom domain.

02

Redirects & caching

Configure viewer protocol policy to redirect HTTP → HTTPS and **Cache behavior (**`default_cache_behavior`**)**

- `allowed_methods = ["GET", "HEAD"]` → Only allow read requests.
- `cached_methods = ["GET", "HEAD"]` → Cache these requests.
- `viewer_protocol_policy = "redirect-to-https"` → Forces users to use HTTPS (secure).

SSL certificate

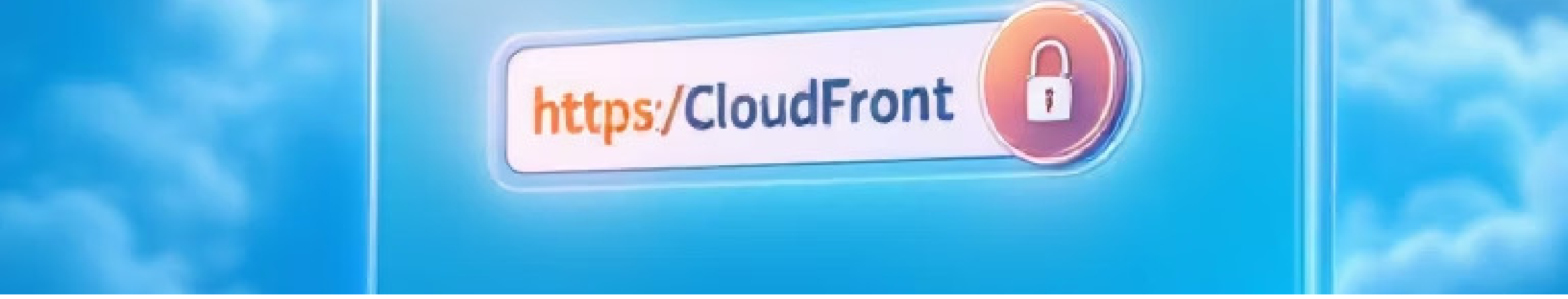
- `viewer_certificate { cloudfront_default_certificate = true }` → Uses CloudFront default HTTPS certificate.

Restrictions

- `geo_restriction { restriction_type = "none" }` → Website accessible from all countries.

Tags

→ Helps identify the resource.



Outputs & Testing

cloudfront url

CloudFront URL

- **What it is:** The **HTTPS domain name** of the CloudFront distribution
- **Purpose:** The public URL users use to access your content through CloudFront — primary global endpoint for testing

s3_url

S3 website endpoint

- **What it is:** The **S3 static website endpoint URL**
- **Purpose:** The direct URL where the S3 bucket content is hosted as a static website

cloudfront id

- **What it is:** The unique **ID of the CloudFront distribution**
- **Purpose:** Used to reference or manage the CloudFront distribution in AWS

```
output "cloudfront_url" {
  value       = "https://${aws_cloudfront_distribution.website_cdn.domain_name}"
  description = "The CloudFront distribution URL for the website"
}

output "s3_url" {
  value       = aws_s3_bucket_website_configuration.website_config.website_endpoint
  description = "The S3 website endpoint URL"
}

output "cloudfront_id" {
  value       = aws_cloudfront_distribution.website_cdn.id
  description = "The CloudFront distribution ID"
```

Testing checklist:

- Verify HTTPS and valid certificate at CloudFront URL
- Confirm content served and cached at edge
- Run terraform output to retrieve URLs after apply

Reflection

What we learned

Deploying a static site with S3 + CloudFront provides a cost-effective, globally-distributed delivery model. Terraform enables repeatable infrastructure.

Challenges

Configuring CloudFront TLS, managing origin access (OAI vs public), and Terraform resource dependency ordering were the main hurdles.

Solutions

Used CloudFront's default certificate for quick HTTPS, followed AWS & Terraform docs, and added explicit depends_on or data sources to clarify dependencies.

References

Primary references: https://aws.amazon.com/s3/ • https://aws.amazon.com/cloudfront/ • https://www.terraform.io/