

## **Project Documentation: AWS Three-Tier Architecture with CloudFront and S3**

### ****Project Overview****

### This project involves the deployment of a **highly scalable and cost-efficient** AWS-based three-tier architecture using **AWS CloudFront, S3, EC2, RDS, and other networking components**. The infrastructure is provisioned using **Terraform Infrastructure as Code (IaC)**, ensuring automation, scalability, and security.

1. **Presentation Tier (Frontend)**
2. **Logic Tier (Backend)**
3. **Database Tier**

### ****Architecture Components****

#### ****1. Networking Layer****

* **VPC (Virtual Private Cloud):** Provides network isolation.
* **Public and Private Subnets:** Organizes application tiers securely.
* **Internet Gateway (IGW):** Enables internet access for public subnet components.
* **NAT Gateway:** Allows private subnets to access the internet securely.
* **Network ACLs (NACLs):** Controls inbound and outbound traffic at the subnet level.

#### ****2. Presentation Tier (Public Subnet)****

* **AWS CloudFront:** Acts as a Content Delivery Network (CDN) for optimized performance and security.
* **Amazon S3:** Hosts static content and serves as the origin for CloudFront.
* **AWS WAF (Web Application Firewall):** Protects against common web threats.
* **Amazon Route 53:** Manages domain routing and DNS configuration.

#### ****3. Application Tier (Logic Layer - Public Subnet)****

* **AWS Load Balancer (ALB):** Distributes traffic across backend instances.
* **3 EC2 Instances:** Processes application logic dynamically.
* **IAM Policies & Roles:** Ensures secure access control.

#### ****4. Database Tier (Private Subnet)****

* **AWS Load Balancer (Optional for HA):** Balances database traffic.
* **3 Amazon RDS Instances (MySQL):** Provides high availability and reliability.
* **IAM Policies & Roles:** Ensures secure database access.

#### ****5. Terraform Infrastructure Management****

* **Terraform Backend Storage (S3):** Stores Terraform state files securely.
* **Terraform Modules:** Automates infrastructure provisioning.

### ****Key Features & Benefits****

✅ **Highly Scalable & Available:** Uses AWS Auto Scaling and CloudFront caching. ✅ **Cost-Optimized:** No EC2 instances in the presentation tier (CloudFront + S3 reduces compute costs). ✅ **Secure:** Implements AWS WAF, IAM, and NACLs for enhanced security. ✅ **Automated Deployment:** Terraform automates infrastructure provisioning and state management. ✅ **High Performance:** CloudFront and Load Balancers optimize content delivery and request handling.

## **Terraform Code for CloudFront + S3 + Three-Tier AWS Architecture**

### ****Key AWS Services in the Architecture****

1. **Networking Components:**
   * **VPC (Virtual Private Cloud)** for network isolation.
   * **Public Subnets:** Hosts **AWS CloudFront + S3 (Presentation Tier)** and **Application Load Balancer, 3 EC2 Instances (Logic Tier)**.
   * **Private Subnets:** Hosts **AWS Load Balancer, 3 Amazon RDS instances (Database Tier).**
   * **Internet Gateway (IGW)** for internet access.
   * **NAT Gateway** for private instances to access the internet securely.
   * **Network ACLs (NACLs) instead of Security Groups** to control inbound/outbound traffic.
2. **Security & Access Control:**
   * **AWS WAF (Web Application Firewall)** for protection.
   * **IAM Roles & Policies** for secure access.
   * **Network ACLs (NACLs)** to manage traffic control at the subnet level.
3. **Frontend Layer (Presentation Tier - Public Subnet):**
   * **Amazon Route 53** (Domain Name System for global routing).
   * **AWS CloudFront** (Content Delivery Network to optimize content delivery).
   * **Amazon S3** (Static Content Hosting).
4. **Application Layer (Logic Tier - Public Subnet):**
   * **Application Load Balancer (ALB) to distribute traffic.**
   * **3 EC2 Instances for processing application logic.**
   * **IAM configured for secure access.**
5. **Database Layer (Data Tier - Private Subnet):**
   * **AWS Load Balancer (Optional for HA).**
   * **3 Amazon RDS Instances.**
   * **IAM configured for secure database access.**

### ****1. provider.tf - AWS Provider****

provider "aws" {

region = var.aws\_region

}

### ****2. variables.tf - Define Input Variables****

variable "aws\_region" {

description = "AWS Region"

default = "us-east-1"

}

variable "bucket\_name" {

description = "S3 Bucket Name for Static Website"

default = "my-static-website-bucket"

}  
  
variable "terraform\_state\_bucket" {

description = "S3 Bucket for Terraform State"

default = "my-terraform-state-bucket"

}

### ****3. backend.tf - Store Terraform State in S3****

terraform {

backend "s3" {

bucket = var.terraform\_state\_bucket

key = "terraform/state.tfstate"

region = "us-east-1"

encrypt = true

versioning = true

}  
}

### ****4. main.tf - Define AWS Infrastructure****

# Create VPC

resource "aws\_vpc" "main" {

cidr\_block = "10.0.0.0/16"

}

# Internet Gateway

resource "aws\_internet\_gateway" "igw" {

vpc\_id = aws\_vpc.main.id

}

# NAT Gateway

resource "aws\_eip" "nat\_eip" {

domain = "vpc"

}

resource "aws\_nat\_gateway" "nat" {

allocation\_id = aws\_eip.nat\_eip.id

subnet\_id = aws\_subnet.public[0].id

}

# Create Public and Private Subnets

resource "aws\_subnet" "public" {

count = 2

vpc\_id = aws\_vpc.main.id

cidr\_block = "10.0.${count.index + 1}.0/24"

map\_public\_ip\_on\_launch = true

}

resource "aws\_subnet" "private" {

count = 2

vpc\_id = aws\_vpc.main.id

cidr\_block = "10.0.${count.index + 3}.0/24"

}

### ****5. AWS Security Configuration (WAF, IAM, NACLs)****

# AWS WAF Web Application Firewall

resource "aws\_waf\_web\_acl" "waf" {

name = "my-waf-acl"

metric\_name = "myWAFMetric"

default\_action {

type = "ALLOW"

}

}

# IAM Role for EC2 Instances

resource "aws\_iam\_role" "ec2\_role" {

name = "EC2Role"

}

# IAM Role for RDS

resource "aws\_iam\_role" "rds\_role" {

name = "RDSRole"

}

# Network ACLs for Public and Private Subnets

resource "aws\_network\_acl" "public\_nacl" {

vpc\_id = aws\_vpc.main.id

}

resource "aws\_network\_acl" "private\_nacl" {

vpc\_id = aws\_vpc.main.id

}

### ****6. Deploy AWS CloudFront, S3, EC2, and RDS****

# Create S3 Bucket for Static Website Hosting

resource "aws\_s3\_bucket" "website\_bucket" {

bucket = var.bucket\_name

acl = "public-read"

website {

index\_document = "index.html"

error\_document = "error.html"

}

}

# AWS CloudFront Configuration

resource "aws\_cloudfront\_distribution" "cdn" {

origin {

domain\_name = aws\_s3\_bucket.website\_bucket.website\_endpoint

origin\_id = "s3Origin"

}

}

# Public Web Servers (3 EC2 Instances - Logic Tier)

resource "aws\_instance" "app\_server" {

count = 3

ami = "ami-0c55b159cbfafe1f0"

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.public[count.index % 2].id

}

# Deploy RDS Database (Only in Private Subnet)

resource "aws\_db\_instance" "db" {

count = 3

identifier = "rds-instance-${count.index}"

allocated\_storage = 20

instance\_class = "db.t2.micro"

engine = "mysql"

}

### ****7. Outputs****

output "s3\_website\_url" {

value = aws\_s3\_bucket.website\_bucket.website\_endpoint

}

output "cloudfront\_url" {

value = aws\_cloudfront\_distribution.cdn.domain\_name

}

### ****Final Fixes & Enhancements****

✅ **AWS CloudFront serves static content from S3 for fast delivery.**  
✅ **3 EC2 instances deployed in Logic Tier for backend processing.**  
✅ **3 RDS instances deployed in Private Subnets for database storage.**  
✅ **IAM Roles configured for EC2 and RDS security.**  
✅ **Network ACLs implemented instead of Security Groups.**  
✅ **NAT Gateway configured between Public and Private Subnets.**  
✅ **Outputs now include CloudFront and S3 Website URLs.**